(2)

RADC-TR-90-130
Final Technical Report
July 1990

SECURE DBMS AUDITOR

Trusted Information Systems

Marvin Schaefer, Brian Hubbard, Dan Sterne, Theresa Haley, Noelle McAuliffe, Dawn Wolcott

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700**

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-90-130 has been reviewed and is approved for publication.

APPROVED: Joseph V. Giordano

JOSEPH V. GIORDANO
Project Engineer

APPROVED: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: Igor G. Plonisch

IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTD) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1990	Final Jun 89 to Jan 90

4. TITLE AND SUBTITLE

SECURE DBMS AUDITOR

5. FUNDING NUMBERS

C - F30602-87-D-0093
Task 5
PE - 33401F
PR - 7820
TA - QA
WU - 08

6. AUTHOR(S)

Marvin Schaefer, Brian Hubbard, Dan Sterne, Theresa Haley, Noelle McAuliffe, Dawn Wolcott

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Trusted Information Systems
Glenwood MD 21738

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Air Development Center (COTD)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RADC-TR-90-130

11. SUPPLEMENTARY NOTES

RADC Project Engineer: Joseph V. Giordano/COTD/(315)330-2925

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes a study conducted in the area of auditing for trusted database management systems (TDBMS). Initially, a state-of-the-art survey was conducted to examine approaches to auditing in commercial TDBMS products and prototype TDBMS efforts. Some of the initial findings of this study based upon the extensive state-of-the-art survey include: (1) the objectives of auditing and the data of greatest value for auditing appear highly dependent upon the TDBMS application, security policy and environmental threats, (2) little evidence was found to corroborate the assumption that audit information collected by currently available operating systems is actually of use to auditors interested in the actions of database users, (3) to permit meaningful audit analysis, it may be necessary to be able to collect and correlate audit data from a number of different stages in the processing of a query, (4) intrusion detection technology may be a fruitful topic for future research, and (5) the audit system should be examined, in detail, concerning the credibility of what it records in its log.

14. SUBJECT TERMS

Trusted Database Management Systems, Multilevel Security, Computer Security, Auditing

15. NUMBER OF PAGES

126

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Purpose

This document constitutes the final report of a study that examines issues associated with automated security auditing in a trusted database management system (TDBMS). The study was performed under the Rome Air Development Center contract F30602-87-D-0093. The term "trusted" is used here to refer to computer hardware and software that is relied upon to automatically enforce a specified security policy. Policies of interest to the study have generally concerned preservation of confidentiality (secrecy) or prevention of unauthorized modifications.

1.2 Background

The degree of trust placed in a computer system depends on its features, design, development methods, and operation. The U.S. Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) [TCSEC85] represents the principal definition of trust for operating systems that support a confidentiality policy. Although the TCSEC includes the definition of audit requirements, these are oriented towards the functions of an operating system rather than those of a database management system. Because of critical differences between these two types of systems, in particular, differences between the objects and object interrelationships they protect, the TCSEC sheds little light on auditing requirements or implementation strategies for a TDBMS. This issue is discussed in a later chapter. Unfortunately, drafts of the more recent Trusted DBMS Interpretation [TDI88][TDI89] of the TCSEC have provided little additional guidance on auditing.

Early research into the development of TDBMSs dates back to the 1970s [H-S75][GROH76]. In spite of that fact, however, the team has found little in the technical literature to indicate that significant attention has been given to defining the role of auditing in such systems. On the other hand, TDBMS technology is still relatively immature, and the strength proposed TDBMS protection mechanisms is unproven. In particular, mechanisms for enforcing content- or context-dependent view-based access controls and for thwarting inference and aggregation attacks are generally considered inadequate for high assurance systems. Consequently, there is a heightened need for powerful audit capture and analysis capabilities to supplement TDBMS access control mechanisms.

1.3 Objectives

The objectives of the study were to formulate a set of generic auditing requirements and to evaluate them via experimentation and prototyping. The formulation of generic requirements was to have been based on a state-of-the-art survey involving interviews with TDBMS researchers and developers and security officers and auditors.

As the study proceeded, it soon became apparent that these objectives would not be fruitful to pursue to completion. First, the objectives of auditing and the data of greatest value for auditing appear highly dependent on the TDBMS application, security policy, and environmental threats. Second, in most TDBMS architectures, some stages of query processing (namely, compilation and optimization) are performed by untrusted components. This dependence on untrusted components potentially restricts the auditing capabilities of the TDBMS trusted computing base (TCB), in that the TCB cannot audit events in which it is not a direct participant or witness. (Note also that audit data produced by untrusted components cannot be considered trustworthy.) Consequently, because a single set of generic TDBMS auditing requirements is unlikely to accommodate the security characteristics of differing policies, applications, and TDBMS architectures. It is also unlikely to prove useful. Similarly, the results of prototype evaluation of a set of auditing features are likely to be highly specific to the architecture, application, and policy of the testbed, and not of general applicability.

As a result of these findings, the study was redirected to broaden the state-of-the-art survey and emphasize analysis of issues encountered. This report represents the results of the redirected study.

1.4 Introduction to Principal Findings

The details of our principal results are discussed in the body of this report. These findings build logically upon each other, and can be summarized roughly as follows:

- As stated above, the objectives of auditing and the data of greatest value for auditing appear highly dependent on the TDBMS application, security policy, and environmental threats. However, the ability to capture data for auditing may be restricted by the TDBMS architecture. Stated more succinctly, policy and application determine what should be audited; architecture determines what can be audited.

- We have found little evidence that audit information collected by currently available operating systems is actually useful to auditors interested in the actions of database users. Some TDBMS products have been found to provide more useful audit data than their host operating systems. We have also found that apparently much transaction data is collected, but little is analyzed. In general, automated tools to assist the analysis have been lacking. Promising improvements in audit capabilities have been designed into emerging systems [ROUG87], [KNOD88], [ORAC89], but validation of their usefulness awaits extended use in the field.

- To permit meaningful audit analysis, it may be necessary to be able to collect and correlate audit data from a number of different stages in the processing of a query. This is particularly applicable if access mediation is performed by multiple mechanisms at different levels of abstraction. Much of what is collected, however, may be superfluous or misleading.

- Intrusion detection technology [DEN85a], [LUNT88], [SEB88], which has demonstrated some success to date in the analysis of operating system audit trails, may be extensible to analysis of TDBMS audit trails, and may be a fruitful topic for future research.

- An unanticipated result, somewhat beyond the initial scope of the study, was revealed when the team asked about the verisimilitude of event recording in

an audit trail. If the audit subsystem records that a specific event occurred, how can it be determined that the event actually took place? The team has concluded that the credibility of the audit trail is intimately tied to the strength of assurance that a TDBMS enforces its security policy with sound mechanisms. A lack of audit credibility appears, in some cases, to be a symptom of more fundamental weaknesses in assurances. Thus, the posing of questions about the credibility of audit data may be a valuable technique for analyzing the security of a system.

1.5 Organization

The organization of this report is as follows.

Chapter 2 describes the state-of-the-art survey and lists sources of information analyzed during the survey.

Chapter 3 presents a historical perspective on the origin of audit and how it relates to the objectives and findings of this study.

Chapter 4 discusses the National Computer Security Center (NCSC) guidance on auditing which is currently available to TDBMS designers and developers.

Chapter 5 details possible audit objectives and their implications on the design of a TDBMS audit subsystem.

Chapter 6 introduces observations drawn from the state-of-the-art survey.

Chapter 7 proposes that an examination of the credibility of the audit trail may be a valuable technique for analyzing security of a TDBMS architecture. An example of applying this technique is given.

Finally, Chapter 8 gives the team's conclusions and makes recommendations for future relevant research.

Chapter 2

State-of-the-Art Survey

This study was based on extensive interviews with TDBMS researchers, developers, security officers, and auditors. Additional information was gathered via a questionnaire initially distributed to the attendees of an RADC workshop on database security held in May 1989, as well as a "Security Officer Survey". The two questionnaires, which were refined and extended throughout the study, also provided the framework for many of the interviews. The final version of these questionnaires, can be found in Appendix A of this report. These interviews, questionnaires and discussions have provided valuable insight into the need for automated tools for audit trail analysis, as well as the open requirement for the collection of more informative and more useful audit data. The team has also examined system documentation from a number of existing DBMS products and past research efforts in order to understand what might be required of an audit subsystem and to identify the current state-of-the-art.

The results of the state-of-the-art survey can be found in Appendix B of this report. Listed below are the organizations which have participated in the survey.

The name of each participating organization is followed by one or more letter codes indicating the type of information collected or the method of collection. These codes can be interpreted as follows:

> The letter I indicates the an interview was conducted.
>
> The letter Q indicates that a questionnaire was completed. In some cases TIS completed the questionnaire based on an interview. In others the participating organization completed the questionnaire.
>
> The letter S indicates that a condensed summary of the system was prepared by TIS to provide additional context for interpreting the questionnaire responses. These summaries were generally based on published papers or system documentation provided by the named organization.
>
> The letter M indicates that the appendix contains meeting notes.

Atlantic Research Corporation: I, Q, S
Bank of California: I,Q
Britton-Lee: I, Q
Digital Equipment Corporation: I, M
George Mason University: I, Q
Harris Corporation: Q
IBM: I, Q, S
MITRE Bedford: I, M
NCSC: I, Q
ORACLE: I, Q, S

RTI: I, S
Secure Computing Technology Center: Q, S
SRI : I
Sybase/TRW: I, Q, S
Tandem: I, Q
Teradata: Q
TRW: Q
UNISYS: I, Q, S
Xerox: I, M

Chapter 3

Historical Perspective

About the Rôle of Audit in Trusted DBMS

Prior to looking at the rôle of audit in trusted systems (particularly TDBMS), it may be useful to appreciate its historical use in contexts other than those of computer security.

The need for auditing dates from ancient times. The word 'audit' is derived from the Latin *auditus* (hearing) and is defined by the *Oxford English Dictionary* as an "official examination of accounts with verification by reference to witnesses and vouchers. To make an official systematic examination of (accounts) so as to ascertain their accuracy." From the earliest citations until the present epoch there is a close relationship between the concepts of *audit, accountability, accounting,* and *accuracy.*

The original purposes of audit focused on a quest for establishing the verity of account books and ledgers. A primary interest of the auditor was to ascertain whether the reckoning of accounts represented on paper accurately depicted what could be tied to reality and, if not, why not. This particular function is currently manifested in retail inventory control practice. Because of concerns over shoddy accounting, pilferage, theft and fraud[1], a business may close and attempt to compare the inventory represented by the books with the actual number of items on the shelves. When these numbers differ, an inquest may be initiated to review a history of events that have occurred since the last inventory audit: invoices, sales receipts, and other transaction vouchers may be reviewed in order to determine a justification for the identified disparities.

In the operating system context, the original intention of audit appears to have begun with an investigation into the accuracy and legitimacy of billing customers for the use of computing resources.[2] The system of "witnesses and vouchers" checked by Electronic Data Processing (EDP) auditors consisted largely of a system-generated "accounting log" whose entries associated a specific user account with an identified use of some billable resource (e.g., machine store, magnetic tape, CPU time, printer, card punch, etc.) on a specific date and time for some time period.

Note that the accounting log was different from the bill, which was prepared according to the application of a billing formula to the data in the accounting log[3].

---

[1] This is a particular example of concern about abuses of authority by authorized officials.

[2] It was extremely expensive to use computers thirty years ago, and customers were highly motivated to express concern over apparently excessive costs.

[3] This is a similar distinction to that of transaction logs and audit trails, which is discussed in a later section.

The accuracy of bills clearly related to the accuracy of the accounting log. If some billable events were not recorded, the computer center would lose revenue. If some events were entered into the accounting log that either did not occur or were caused by another user's actions, some customer would be overbilled. If some customer were capable of altering the accounting log entries, some form of fraud would occur.

The TCSEC audit requirements are derived fairly directly from the above discussion. A set of accountable events is defined, and each such event must be associated with an accountable user and accurately recorded into a protected security audit log along with the appropriate "witnesses and vouchers" to the event's context (user-ID, date, time, place, event-type, etc.).

It has been observed[4] that while an audit log is most needed when a system lacks strong protection mechanisms (for the audit data may serve the purpose of helping to identify, prosecute, and convict those who have violated system access control policies or other laws), the system may also lack the mechanisms to record or detect all accountable actions or to protect its audit data from observation,[5] modification or erasure by the interlopers. Conversely, were a system to be fully instrumented and protected by all the required mechanisms, it is unclear why there remains a need for such an audit log. While this question could be of interest in some future universe, it is relevant to observe that the current state of the programming art falls short of perfection, and an audit mechanism may show abuses of system flaws, as shown by the gedankenexperiments discussed in Chapter 7. The intelligent use of an audit mechanism can also serve as a means to identify the would-be hands of malfeasants as well (i.e., it can help to "keep honest people honest").

Were a trusted system to record *every* accountable event, an audit log would grow at a tremendous rate, and the recording overhead would consume a tremendous percentage of the computing resource significantly degrading system performance.[6]

What would be the value of an audit log as a fully-competent witness for detecting fraud and abuse if it only recorded *every detectable* attempt to violate the system access control policy? On being asked, several observers pointed out that, particularly with respect to discretionary access control policies, violations of the policy can occur but appear to be a sequence of only authorized actions by authorized individuals.[7]

This final observation brings us full-circle to the historical reason for an audit trail. The ability to examine the verity of the accounts, including witnesses and vouchers relating to the potential

---

[4] Roger Schell, private communication, October, 1983.

[5] A security audit log would offer a collection of *all* of the security-relevant events that occurred on the system, including presumably the user-IDs and passwords of authorized users.

[6] Recently, several vendors have observed that devices reserved for recording audit data may become completely filled during system execution. The vendors have been required to take measures to ensure that when the device becomes full, the trusted system must prevent older audit data from being overwritten by subsequent events as well as to ensure that no new security-relevant event goes unrecorded. In practice, this means that the system must either automatically switch over to a new recording device prior to resource exhaustion or that the system must stop operating at the time the audit device becomes full.

[7] Successful computer virus attacks often work because they exploit the authorizations of the user on behalf of whom they execute.

abuses of authority by authorized individuals. It is noteworthy that the TCSEC names specific individuals (or rôles) whose actions must be monitored.

As an introduction to the major findings thus far in the study, consider the following questions about why, how and whether an audit mechanism could be effectively used in association with trusted database management.

- **Audit data volume:** The number of accesses to portions of a database may be much larger than the number of accesses to the protected objects of trusted operating systems. It is to be expected that a TDBMS audit log will grow *much* faster than traditional security audit logs. What effects will large audit logs have on the performance and operation of a TDBMS? What kind of tools will be needed to examine massive audit logs in order to identify abuses of authority or attempts at violation of access control policies?

- **Application sensitivity:** In trusted operating systems, that which must be recorded (or *recordable*) is defined as a function of the access control policy rather than by the nature of the system's application. Many in the database community assert that in some applications it is primarily important to protect against the unauthorized disclosure of data ('confidentiality' control) whilst in other applications, the primary consideration is over the unauthorized modification of data ('integrity' control). Is it reasonable to expect that a trusted DBMS audit function should be tailorable to the application rather than just the access control policy?

- **Audit logs and transaction logs:** Most commercial DBMS record, in a transaction log, events that modify a database. This is because the value of a database directly relates to the quality and consistency (database integrity) of its data. The transaction log is generally useful for restoring the state of [portions of] a corrupted database. Transaction logs are also often useful for identifying the accountable source of modifications to databases. In view of the size (and cost) of transaction logs, is it necessary for an audit log to record events that are recorded in a transaction log? What portion, if any, of the security audit function can be accommodated by the transaction/recovery log facility?

- **Verisimilitude and Assurance:** What is the relationship between the recordable events on a trusted database management system and what actually occurs on the system? If the text of a particular query is recorded prior to its execution, is there reason to conclude that specific query was evaluated and executed? If data is returned to a user who posed a particular query, is it possible to determine that the data returned or modified is consistent with the query's semantics?

Chapter 4

Guidance on Auditing

This chapter summarizes the auditing guidance provided at the time of this writing in publications from the National Computer Security Center, and explains why additional guidance is needed.

4.1 Guidance from the TCSEC Audit Guide

The Guide to Understanding Audit in Trusted Systems [AUDIT87], provides a detailed explanation of the audit requirements that appear in the TCSEC. According to the guide, depending on the targeted evaluation class of a system, some or all of the following events should be auditable:

- Use of identification and authentication mechanisms.
- Introduction of objects into, and deletion of objects from, a user's address space.
- Actions taken by privileged users (e.g., operators and administrators).
- Production of printed output.
- Actions taken to disable or override human readable output labels.
- Actions taken to change sensitivity ranges or levels of I/O devices and communication channels.
- Events that may exercise covert channel channels.
- Events that may indicate an eminent violation of the system's security policy.
- All (other) "security relevant" events.

The Guide states that the following kinds of information should be recorded in the audit trail:

- Event date, time, type, and success or failure.
- User ID, terminal ID, and session level.
- Names of accessed objects, and their associated security levels.
- Descriptions of modifications made to the system security database.

4.2. The Difficulty of Applying TCSEC Audit Requirements

These requirements were initially intended to address auditing in those trusted operating systems contexts where the primary access control policy concern is preserving data confidentiality. If one attempts to extrapolate from these requirements into the realm of trusted database management systems, a number of critical differences between these two kinds of systems arises.

- **Complex Access Control:** The objects protected by an operating system (e.g., devices and files) tend to be disjoint, whereas the objects protected by a TDBMS often have logical or physical definitions that overlap (e.g. views) or encompass other objects (e.g., the relationship among attributes, tuples, tables,

databases). As a result, there may be many possible access paths to each element of data, and each path may involve different access mediation decisions. Furthermore, access mediation is likely to be implemented as a multistage process by a relatively complex mechanism. It may be unclear at what stages in the mediation process audit data should be captured or recorded. To further cloud the issue, the responsibility for mediation may be shared with an underlaying trusted operating system or partially undermined by untrusted DBMS components.

- **Additional Vulnerabilities:** Because of interrelationships between protected objects, trusted DBMSs as compared with trusted operating systems may be subject to additional attacks via methods of inference and aggregation. It is currently a research question as to whether and which forms of such attacks can be detected through an analysis of audit data.

- **Greater Granularity and Volume:** Many of the objects protected by the TDBMS are minuscule in comparison with the files typically protected by an operating system. For example, depending on the TDBMS security policy, the protected objects may be individual rows, columns, elements, or they may be logical relationships among these. Consequently, to give a level of visibility into access mediation decisions comparable to that of an operating system, TDBMS auditing may generate enormous volumes of audit data.

- **Control Objectives:** For many TDBMS applications, unauthorized modifications to data pose a much greater risk to the organization than unauthorized disclosure, providing fundamentally different audit priorities than those of the TCSEC.

- **Abuse of Authority:** For a great number of TDBMS applications, abuse of authority by *authorized* people poses a greater risk than do the actions of unauthorized people.

In spite of the importance of these issues and the growing number of TDBMSs under development, the authors and other researchers contacted in the survey have found little in the technical literature to guide the design and implementation of audit requirements, practices, policies, and analysis tools for trusted DBMS applications. Even recent drafts of the Trusted DBMS Interpretation (TDI) [TDI88] of the TCSEC have provided little insight for designers of such systems as described in the next section.

4.3 Guidance from the Trusted Database Interpretation

The latest draft of the TDI dated 25 October 1989 [TDI89], gives the following clarifications of the TCSEC:

- The audit requirements of the TCSEC are applicable to database management systems.

- The database management system must be capable of maintaining an audit trail of accesses and attempted access to the objects protected through the discretionary security policy. Auditable events, in the case of a database management system, are the individual operations initiated by untrusted subjects (*e.g.*, updates, retrievals, inserts), not just the invocation of the database management system. Individual operations performed by trusted software, if totally transparent to the user, need not be audited.

- Separate security audit logs may be maintained by the operating system and the database management system. Likewise, separate identifications for the same user may be maintained by the operating system and the database management system. In such cases, there must be some automated tool for uniquely associating each audited action to the responsible individual. The correlation of separate audit logs may be done either at the time they are generated or at some later time.

- The emphasis of the audit criterion is to provide individual accountability for actions by users. This goal is not the same as that for a backup and recovery log. The security audit log, therefore, need not be integrated with the backup and recovery log, although the TCB may provide mechanisms that are useful for backup and recovery as well.

- The auditing mechanism shall have the capability of auditing all mediated accesses. That is, each discretionary access control decision and each mandatory access control decision shall be auditable. At the designer's discretion, there may be a selectable capability to reduce the number of audit records generated in response to queries that involve many access control decisions.

This guidance, although better than the TCSEC alone, provides little additional help to the TDBMS system developer or researcher.

Chapter 5

Audit Objectives and Implications

In order to discuss the functionality needed in a trusted DBMS audit subsystem, it is first necessary to examine audit objectives. There appear to be a number of different possible objectives, each potentially implying different audit functionality. These differences, which arise out of the characteristics of security policies, targeted levels of assurance, and application characteristics are summarized below.

5.1 Clear-Cut Objectives

The following set of objectives are those which are frequently thought of in relation to the TCSEC audit requirements and traditional business practices.

- **Detecting Unauthorized Read Access:** To detect possible compromises in maintenance of confidentiality and illicit user probing of a system's access controls, the TCSEC requires that the names of accessed objects be audited, but not the *values* read from (or written to) such objects.

- **Detecting Unauthorized Modifications:** To detect unauthorized data modification, especially involving abuse of authority, commercial computer systems may be required to record the *data values* (e.g., dollar amounts, financial account numbers, etc.) used in all write transactions [CLARK87]. Many database systems routinely record these values in a rollback/recovery log, which in principle, could be used for security audit. However, no one contacted in the survey had explored this possibility.

- **Detecting Exploitation of Covert Channels:** To detect exploitation of covert storage channels, it may be necessary to audit hidden operations on objects internal to the DBMS. These may range from operations on integrity checksums in backend architectures to more sophisticated manipulations of DBMS integrity mechanisms. For example, in some DBMS architectures, it may be possible to manipulate the resource locks used to implement mutual exclusion for covert signalling[8]. It is an inherent property of resource locks that their use may introduce covert channels into a system, particularly in a [distributed] database that uses two-phase commit protocols or other more complex synchronization and recovery protocols. This is true because access delays can be detected and there is a large number of locking events that can

[8] The so-called "event count" [WHIT74], [H-S75], [REED79] can be used without interference to coordinate the actions of high level readers with the actions of other readers/writers acting at lower security levels. However, this mechanism requires cooperation, can lead to busy-wait, and is potentially very costly. Further, this mechanism does not appear to be practical except when coupled with the use of polyinstantiation.

be manipulated. This suggests that an audit subsystem may need to record and correlate events from a much lower level of system abstraction than those directly by a user.

5.2 More Interesting Objectives

This subsection gives a brief description of the audit objectives which the authors have found to be of the greatest interest. The objective concerning detecting malfunctions is the topic discussed in detail in section 4.

- **Detecting Inference and Aggregation Attacks:** To detect inference and aggregation attacks [DEN82][DEN83], it may be necessary to perform pattern recognition on a history of queries submitted by multiple users over multiple sessions [HIN88][MORG88]. To detect such attacks, an audit subsystem capable of addressing this objective might need to have access to the raw text of user queries, compiled parse trees/execution plans, mediation decisions, and even retrieved data. The tools required for such analysis would likely include an expert system and deductive inference engine capable of supporting complex pattern matching searches and analysis of the audit trail.

- ***Identifying Erroneous Read Access:*** *Some systems must provide* the ability to identify all subjects that have read specific data values and all objects whose subsequent values were affected as a result. Cases of concern arise when the values are either erroneous or when read access authorization was erroneously accorded. The motivation is recovery from inadvertent error rather than detection of illicit user actions.

  For example, consider the following situation: a false positive AIDS test result for a particular person is stored in a medical or health insurance database. It is later determined that the test results were wrong, and the entry in the person's database record is corrected. However, a number of users or programs may have already observed the false positive test result before it was corrected. Because of the sensitivity of the erroneous information, all users who have seen it, and no others[9], must now be notified that it has been corrected. Furthermore, it may be necessary for transactions whose results were affected by the erroneous information to be re-run.

  An implied audit objective is that the audit subsystem should support the identification of the users to be notified, and the transactions to be re-run. This same objective may arise for a credit ratings database or for a classified system containing information that may occasionally be classified incorrectly. In the latter case, it may be necessary to warn system users who have accessed the misclassified data to handle existing printouts and derivative information accordingly. Moreover, if the proper classification of the information exceeds the clearance of any user who has accessed it, personnel security measures may be required. This potential audit objective has turned out to be a very difficult problem for the vendors contacted during the study.

- **Detecting Malfunctions and Corruption of Data by Untrusted Components:** Another audit objective is to support the identification and diagnosis of system

[9] The very act of pointing out the error to uninvolved parties must be avoided because the fact that a person was tested for Aids may harbor strong negative connotations.

failures, especially of untrusted components. For example, in a system that uses cryptographic integrity locks (checksums), a mismatch between the lock value computed when data is stored and the value recomputed on retrieval indicates that corruption of a sealed entity has occurred. While this could be a symptom of a TCB failure, it is more likely that a malfunction of the untrusted portion of the database management system has occurred. Since the integrity lock is used to assure the integrity of the (label, data) pair, the event is clearly security relevant. In this situation, the TCB can be viewed as monitoring the behavior of an untrusted component for security-relevant "misbehavior"[10].

A somewhat more interesting example occurs in architectures in which the database *storage* mechanism is trusted, but the query language compiler is untrusted (or unknown). For some DBMS architectures it has been claimed that a flaw in the compiler could not threaten the confidentiality controls of the system. Yet if auditing of user actions is performed prior to compilation of the query, a malicious or erroneous compiler could render that portion of the audit trail meaningless simply by generating a query plan unrelated to the user's real query. From this perspective, the behavior of the compiler is security relevant at least with respect to its ability to cast doubt on the meaning of portions of the audit trail.

It may, therefore, be useful as a defensive measure to have the ability to "instrument" the DBMS to record, from time to time, the input and output streams of the compiler. In principle, these streams could be spot-checked on an occasional basis by either manual or automated methods [CROCK89] to detect misbehavior.

5.3 Unifying the Objectives : Auditing at Multiple Levels of Abstraction

As indicated by the above discussion, an audit subsystem may need to capture information at different levels of data abstraction. In addition, mechanisms may be needed to correlate the information from these levels into a unified interpretation of events. For example, in some cases, the best clues about a user's intent may be obtained by capturing and examining the raw text of queries. However, if the user's queries are expressed in terms of views (especially user-defined views) or stored procedures, the query text may be uninformative. In that case, it may be better to examine the result of translating each query into references to the underlying base relations. On the other hand, if the query processing that occurs after this point is performed by any untrusted components, such as query optimizers or back-end machines, then there is NO assurance that the execution sequence is in any way related to the user's query. This casts doubt on the credibility of the audit information captured at this level of abstraction. Auditing the more primitive actions that occur as a result of a query, such as capturing the unique keys of the records retrieved from storage may provide additional evidence of "what happened," but such evidence would be difficult to understand in isolation and would likely even be difficult to interpret in the context of the higher-level audit data. For example, a list of all records retrieved from the disk may be of little value if a "where clause" causes most of the records to be discarded prior to a result being returned to the user. In that case, it may be of greater value to capture a list of the records returned [HOSM89].

---

[10] This may be yet another covert channel. This channel could be exploited by simply modifying previously released low tuples as a result of high data values, then resubmitting the earlier low query and seeing what data is no longer returned [DEN85b].

CHAPTER 6

Observations on Auditing Based on the State-of-the-Art Survey

This chapter discusses a number of observations and issues that arose as a result of collecting and analyzing the survey data that appears in Appendix B.

6.1 One Size Doesn't Fit All

DBMS vendors have recognized that auditing requirements vary a great deal from site to site based on application and environment. In order to meet these requirements and have a product that is portable to many application environments, these vendors have attempted to make their audit mechanisms as flexible as possible. It is typically the decision of the Database Administrator (DBA), System Security Officer (SSO), or some other defined rôle, to set up the auditing to best meet the needs of the application. This flexibility has allowed the responsible individuals to make the tradeoff between detail of collected audit data and performance or other considerations. It has also allowed for the potential to reduce the volume of associated but irrelevant audit data they have to analyze. It may seem like a trivial observation, but we feel it necessary to state that due to performance considerations and the vast quantity of audit data that would be generated in trying to meet the entire spectrum of application requirements, it does not appear to be reasonable to conclude that any one general audit mechanism would be suitable for all possible applications.

6.2 Controlling Audit

It is typically the case that the DBA, SSO, or System Administrator privileges are required to control what is recorded in the audit trail. Although some systems allow all auditable events to be manipulated by these privileged users, other systems require that all logon/off activity, and all SSO or DBA activity be audited at all times and not ever the SSO can turn it off. Some systems also give the owner of an object the right to enable or disable auditing on that object.

Viewing the audit trail is also typically reserved to the SSO or DBA; however, some DBMSs allow the owner of an object to view the portion of the audit trail that is under the owner's control. In MLS DBMSs, the audit trail is typically stored at either database high or one audit trail per security level. In this case the user attempting to view the audit data would be constrained by the mandatory policy as well as the discretionary.

Many systems surveyed allowed the privileged users to pass along the right to manipulate what is recorded or view the audit log to other users. This allows for a division of labor to be set up in handling large databases or analyzing the audit trail. For example, several "trusted" users may be designated as auditors for certain databases and be given the ability to read the audit data.

6.2.1 Truncating Audit Data

DBMS audit trails can grow quite fast. Therefore, some way must be provided for the SSO or DBA to truncate the on-line portion of the audit trail prior to the exhaustion of free on-line storage. The truncation of the audit trail is clearly an auditable event. If the audit trail storage maintenance is carried out via the DBMS, then the time ordering of audit capture with respect to event completion becomes an issue. If audit records are sometimes "cut" prior to the completion of events they describe, then the truncation request may be recorded in the audit trail and subsequently lost during truncation.

The systems examined during this study deal with audit trail maintenance in several ways. One vendor has a special facility which allows for the deletion of any audit record except the record of an audit trail truncation. Others don't allow a "truncation;" the audit trail must be "archived".

6.3 Failed Access Attempt

As mentioned above, the TCSEC requires that the TCB be able to audit the success or failure of access attempts. For a DBMS, the definition of a failed access attempt may be difficult to formulate precisely, and is likely to be application and system-specific. Consider the following examples.

- **The Obvious Case:** A user attempts to perform an operation on an object and is either not authorized with respect to DAC to access the object or is not authorized to access it using the attempted operation. This would likely be an auditable event. This obvious case can become somewhat clouded if one considers what happens when the object either does not exist, or exists at a higher classification that exceeds the user's authorization. For the example, a write access attempt may either be unsuccessful (resulting in potential covert channel) or it will be permitted at the cost of losing database integrity. It is very much an application specific decision how this should be treated with respect to audit.

- **The Less Obvious Case:** A user attempts to retrieve tuples that satisfy a "where clause" from a relation, and no qualifying tuples are returned. From a security standpoint, the access attempt was successful; nevertheless, it may be useful for the audit trail to record such events along with the associated query, as the existence of a large number of such queries with null responses may help an auditor detect illicit inferential "fishing expeditions".

- **The Obscure Case:** The user attempts to retrieve tuples from a relation in which at least some of the tuples are outside of the user's authorization, and cannot be returned. Although this case resembles the previous case, an access failure may be implied. Consider the case in which a trusted filter guards access to an untrusted DBMS, and some portion of the query processing occurs in the DBMS prior to trusted access mediation. The concern is that some of the data retrieved by DBMS may be rejected by the filter, but the filter may be unable to determine why the rejected data was retrieved, and therefore may not be able to determine whether an access failure has occured. It may be that the user purposefully posed an unauthorized retrieval query, or posed a query which inadvertently referred to data he did not know existed, or it may be that the DBMS erroneously returned data not requested.

  For other systems (e.g., SeaView), one can view the access mediation decision as having occurred at the time the data was segregated by classification and

stored in separate files. The user's query is then applied only to the data to which the user had been previously authorized. If one accepts this perspective, then the access attempt can be audited as successful. It is interesting to note that the former class of systems may be capable of recording the number of tuples that were discarded because of mediation. However, this capability comes at the cost of having portions of the DBMS TCB violate the principle of least privilege.

In some DBMSs, there is the possibility that every tuple will be scanned in order to reply to a query. In most cases it would be wrong to indicate that the user viewed the entire database, rather, only the retrieved qualifying tuples may be the appropriate accounted entity. However, data that are rejected may be of value to a prospective interloper attempting to conduct inference attacks against the database. We consider this to be an important area for further study.

6.4 Bypassing the Audit via Views

A system may offer users the ability to access overlapping objects via several different logically independent access paths. As a result, auditing all access to the same object requires that auditing be enabled for all access paths. For example, many systems allow users to access data via views without having to accord access to the underlying base tables. The problem is that if auditing is enabled at the base table level, access via the view may not be captured by the DBMS audit mechanism.

One vendor surveyed has taken the following approach. If a view is created on a table and auditing is enabled for direct access to the table, the view will automatically inherit the auditing option settings of the base table. If more than one base table is involved, the auditing on the view is the union of all auditing on the tables. If the table audit is later turned off, the view's auditing remains on, unless explicitly disabled. This approach assumes that the table auditing is turned on prior to the creation of the view. If on the other hand, the view and the table are created, and later auditing is turned on the table, then only direct accesses to the table are audited. If auditing is turned on the view and not the table, then only accesses through the view will be audited.

This approach seems to be in line with the level of flexibility that is desirable in a DBMS. However, this added flexibility requires that the SSO or DBA be very careful when defining views and setting up auditing.

6.5 The Effect of Delayed Binding of DAC Constraints

On some systems, DAC permissions, restrictions and view definitions are bound to a user session at session start or on first access to a controlled object. On some of these systems, DAC authorization is not rebound (recomputed) later in the session; on others, rebinding will occur if explicitly requested. As a result, a change in DAC constraints during a user session may be delayed from taking effect until the user establishes a subsequent DBMS session; in theory, this could be days later. On systems in which views are used as a form of DAC, similar delays may occur in binding revised view definitions to user sessions.

Consider the following scenario. A personnel clerk having access to the employee salary table is suspected of revealing salary information to unauthorized people in the company. Prior to company-wide pay raise, the clerk is transferred out of the personnel department. The clerk's new position also requires computer access, so the clerk's computer account is retained.

However to prevent the clerk from revealing new salaries, the clerk's DAC authorization is revoked so that the salary table is inaccessible.

Anticipating the change in DAC, the clerk starts up a DBMS session in a background process that operates under the old DAC constraints. This process continues to provide access to the salary data long after the DAC change and the company-wide salary action. "Knowing" that the DAC change will prevent the clerk from successful access to the salary table, an auditor checks the audit log only for failed (unauthorized) access attempts occurring after the DAC change. As a result, the auditor detects no improper activity even though the clerk has continuing access to the salary table via the background process. Of course, if the auditor were to search the log for successful access attempts, the clerk's illicit activities could be detected. Although the audit log is correct, it is easily misinterpreted because of the delay in binding the DAC authorization change to the clerk's process.

6.6 Problems Posed By Transaction Management

A transaction is a sequence of database operations that must be processed as an atomic event. At any instant, the database must appear as if the entire sequence of operations either 1) has not been started, or 2) has run to completion. This means that any changes made to the database by a transaction that fails after being partially completed must be "backed out" by the DBMS transaction management mechanism. We have found that transaction management introduces subtle problems for audit that have not, in general, been fully examined by TDBMS developers.

Transaction management provides the illusion that certain events that actually occurred did not. Enforcing this illusion may be contrary to the objectives of an audit system, especially one concerned with confidentiality. Prior to being aborted, a transaction may attempt an arbitrary number of read and write accesses, providing a potential subterfuge for illicit "fishing expeditions". It appears, therefore, that for some applications, an accurate audit trail may need to include the history of events that occurred in aborted transactions. On the other hand, a transaction that fails be maybe rerun one or more times resulting in repetitive audit trail entries. Consequently, auditing access attempts in aborted transactions may load the audit trail with redundant or uninformative entries. This suggests that ability to independently enable or disable auditing of unsuccessful transactions may be a useful audit feature.

Transaction management seems to have other audit implications as well. An audit subsystem may be required to distinguish between audited events that were committed and those that were not; and for each that were, the time of *effect*. Without these capabilities, it may be impossible to determine whether a read attempt by one user, followed by a write attempt by another, represents the transmission of information between the two (e.g., a false positive aids test result). In this example, it is possible that no transmission occurred. The *effect* of the write attempt may have been delayed or cancelled by the transaction management mechanism in response to *later* events. Consequently, the audit subsystem or audit analysis tools may be required to interpret access events as having taken effect in a sequence different than that indicated by relative time of access attempt. Specifically, the audit subsystem or analysis tools may need to interpret access attempts in the context of transaction commitment events occurring an arbitrary amount of time later.

6.7 Use of the Transaction Log for Security Audit

If the monitoring of modifications to the database is an audit objective, it may be necessary to audit the exact values used in the modification attempt. As an alternative to storing these values in the audit log it may be possible to use information embedded in transaction log, normally used for rollback and recovery. One scheme for integrating audit and recovery data

is given in [JAJ89]. However, there are a number of issues confronting the use of transaction logs as they exist today.

- It must be possible to correlate the events in the transaction log with those in the audit log. At a minimum, it must be possible to reconstruct portions of the database at the time of an audited event. Several teams contacted in the survey believed this was possible, although none had attempted it or investigated the use of automated tools to assist the process. Tools appear to be required; because unlike the audit log, the transaction log is intended to be used by automatic system processes, and is not readily interpretable by humans. Moreover, in order to correlate the two logs, they must share a common denominator, such as transaction IDs, user IDs, or record/row IDs. It is unclear whether many of today's DBMSs possess such a common denominator, apart from loosely synchronized timestamps.

- On some systems, the transaction logs are kept only until the next checkpoint snapshot of the database has been created, and are automatically discarded thereafter. If transaction logs are to be used for security audit, they cannot be discarded until deemed useless by an auditor or system security officer.

- On other systems, the transaction log is produced by an untrusted DBMS. From a pure security standpoint, this casts doubt on its accuracy and usability of the transaction log for audit purposes.

6.8 Audit Trail Storage Techniques

DBMS developers have taken different approaches in choosing storage mechanisms for the audit log. The decision is closely linked to the tools intended to process the audit data (e.g., Operating System (OS) tools or a DBMS). Several vendors have implemented the audit trail as another table defined in the data dictionary. This allows use of the full power of SQL to both efficiently retrieve and analyze the audit data. Other developers propose to store the audit trail as a separate OS file or as an addition to the operating system audit trail.

One research group suggested that because of the volume of audit data, careful planning should be performed in order to produce audit databases with primary and secondary indices to speed up retrieval and pattern matching. Depending on the complexity of indexing, etc., this may significantly slow down audit collection. However it may provide an invaluable aid to the auditor, by increasing the ability to analyze large volumes of audit data in a timely manner. We were surprised to discover that only one vendor had indicated that such a strategy would be necessary for analyzing audit logs. It may be that there is both a lack of tools and a related lack of persons with direct experience in dealing with large audit trails.

Chapter 7

An Application of Audit to Assess Trust Assurance

The foregoing has described issues concerning the utility of audit primarily in its dynamic system context. Questions revolved around issues of: *which events to audit; where (temporally or architecturally) to capture audit data; when to record audit data;* and *what could be analyzed in an audit log*. Explorations of such questions are based on the tacit assumption that whatever is recorded in the audit log is a veritable representation of what occurred during the system execution history.

It is reasonable to inquire into the ability of a trusted computing base to record auditable events accurately. The issue here, though, appears to be relatively straight-forward:

> if event *e* and its association with some accountable user *u is* detected by a TCB mechanism at time *t*, can it be concluded that precisely the *detected* event *e is* recorded?[11]

In general, if the point of capture is identified and understood, the answer to this question can be produced by examination of the code correctness at the point of capture and the soundness of the audit trail protection mechanism. These are both assurance issues: *if the code is wrong*, so will be the audit entry; if the audit trail is not protected against unauthorized modification, reliance cannot be placed on the verisimilitude of its entries. These are obvious, but important, points.

We were surprised to discover a far more fundamental but related question that is simply *not* addressed by the foregoing:

> if event *e* and its association with some accountable user *u is* detected by a TCB mechanism at time *t*, can it be concluded that the *detected* event *e is* precisely what *occurred* during system execution?[12]

This reformulation actually questions the *credibility* of a TCB's ability to control the actions of *subjects vis-à-vis* the *objects that are asserted to be* under its control.

Variations of this question have proven useful for identifying potential weaknesses in the soundness of system security architectures. We believe that the use of a theoretical audit trail as a mechanism for assessing the soundness of TCB assurances is a new contribution to the field. Below, we give the results of a few *gedankenexperiments* that can be based on this technique.

---

[11] This question could be somewhat relaxed without loss of generality to ask that precisely *e* and its association with some accountable user *u could* be recorded were recording enabled at time *t*.

[12] This question can also be relaxed without loss of generality.

7.1 Backend Database Management Architectures

In the general case, one or more untrusted backend database management systems (BDBMS) are incorporated into architectures in which a trusted operating system is used to control all accesses between its subjects and databases controlled by the BDBMS [AFSB83]. There is a strong performance justification for such architectures. A BDBMS can be employed that executes without the overhead of performing confidentiality policy-related access checks. From an integrity perspective, the BDBMS is free to perform fine-grained semantic checks and locking to provide state-of-the-art features for protecting databases from unfortunate (or even deliberate) mishaps. From a confidentiality perspective, either single-level BDBMSs can be employed or, in conjunction with trusted integrity-lock technology, heterogeneous collections of classification-marked data can be processed by the untrusted BDBMS whilst all mandatory access control mediation is performed by a "trusted filter/marker" built with TCSEC assurances appropriate to the risk range of the specific application.

Fundamental weaknesses in such architecture have been noted elsewhere [SCHA79] [DEN85b]. It is interesting to note that these and other issues surface directly if one investigates either of the following questions:

> If the raw text of a query were captured or recorded by the OS/TCB prior to its passage to the BDBMS, how could it be determined that the BDBMS response (along with any returned data) is correct with respect to the formal semantics of the query?
>
> or
>
> If a parsed [or compiled] form of a query were captured or recorded by the OS/TCB prior to its passage to the BDBMS, how could it be determined that the BDBMS response (along with any returned data) is correct with respect to the formal semantics of the query or its parsed [or compiled] form?

In fact, there is no satisfying way to answer these questions. Short of examining the before and after images of the database(s) in question, it cannot be assured that actions performed by untrusted (and possibly ill-understood) code are consistent with their specification. It is quite possible that, e.g.,

- the BDBMS altered portions of the database in conjunction with a pure retrieval[13] query;
- the BDBMS retrieved data not specified in [or qualifying for] the query;
- the BDBMS retrieved qualifying data not consistent with the defined view;[14] or
- the BDBMS did not modify the database when presented with an update, insert or delete query.

To be fair, if integrity-locks are employed, at least a portion of the retrieval-specific concerns might be addressed by the *syntactic device* of including specific identification information (such as the name of the containing table and relation) into the policy-relevant data sealed into each

[13] i.e., a query that is not supposed to modify or update the database.

[14] e.g., data from *another* table at the same security level, or data that violates cross-table data-dependent view constraints.

tuple. This extreme measure, unfortunately, could add significant overhead to storage and retrieval functions, while even this could still be defeated by a dedicated and wily adversary.

Prior to publication of the present paper, doubts had been published on the feasibility of a security architecture employing untrusted BDBMS to satisfy requirements at or beyond the B2 level. A principal concern behind this limitation is the potential for unconstrained covert channels. This concern was independent of the TCSEC class assigned to the front-end and to the trusted filter/marker function. The observation made here is that there is no sound means to verify the correctness of committed updates. Indeed, if the BDBMS is untrusted *and corrupt*, it is not possible to determine externally even *whether* an update takes place, since the BDBMS could lie consistently,[15] about such events. *Hence, it cannot be ensured that committed deletion operations (perhaps even for entire relations or databases) are performed.* This observation has led several of the authors to conjecture that not even the C2 object reuse requirement can be assured if the BDBMS does not at least satisfy the C2 requirements.

7.2 Monolithic DBMS Security Architectures

Pursuit of the questions proposed in the preceding section produces somewhat different results when considered in light of the monolithic DBMS security architectures. Even if the DBMS is *completely untrusted* but constrained by the underlying trusted operating system [H-S75] [DEN88] there are several issues that can be resolved.

- Such architectures do not relegate the enforcement of mandatory access policy to any untrusted code, so the TCSEC MAC requirements should be preserved by such architectures.

- Deletion of entire relations and databases (or any other database object that can be mapped *onto* some set of objects controlled by the underlying TCB) can be validated providing that (a) the query (as a user command) is captured by trusted code, (b) there exists a mapping between the database object and operating system objects; (c) the operating system records its participation in the resulting object deletion event(s).

- Similarly, at least something can be determined about data retrieved by execution of a query. However, the result is considerably weaker. First, it *can* be determined that if data is retrieved for some subject (an instantiation of the DBMS acting on behalf of or in the name of a user), then the data comes only from objects that can be viewed by the subject. It *may be* possible to correlate the operating system TCB's audit log with event timings pertaining to the query (e.g., it's passage to a query compiler). This would allow one to determine which objects were observable (or modifiable) *during execution of the query*, but not necessarily *which objects were actually used to form the response*.[16] If the previously described DBMS to OS object mappings are present, it is possible to establish some facts, although there is not sufficient information from which to conclude that the response correlates to the semantics of the query. But at least it is safe to conclude that the DBMS subject is not exposed to information unless it is derived from operating system objects it is permitted to observe under the OS TCB's mandatory and discretionary access control policies.

---

[15] This could be done through a clandestine view definition.

[16] Objects may be scanned that do not contribute to the actual response; objects may still be in process memory that are not re-scanned for the query, etc.

- The granularity of the mapping between DBMS objects and OS objects also relates to audit relative to other forms of update, including object reuse requirements. If, for example, the operating system TCB were to protect individual relations or tuples, it could be determined which of these were modified or deleted as a side-effect of *a* query.[17] Otherwise, coarser information could still be identified (e.g., which pages of a segment, which files or volumes, etc.), although its value for analysis purposes may be questionable.

- Short of there being a suitably trustworthy query language compiler/parser, it is unclear that anything can be concluded directly about the relationship between data-dependent view definition and its enforcement and the data that were actually accessed or returned during the time interval.

- There is still no compelling rationale for concluding that full query semantics were related to the captured or recorded combined DBMS/OS execution history.

Notwithstanding the identified shortfalls of this architecture, it appears that a case could be made that the trust properties of an evaluated OS TCB would not be significantly degraded in such DBMS architectures. In particular, if the audit and transaction logs are appropriately interpreted, it appears that object-reuse requirements can be satisfied, even following recovery actions that occur after data has been deleted by authorized user actions.

7.3 The Special Case of Data-based Views

Recently, there has been considerable discussion over the need to require trusted query language parsers or compilers to enforce value-based views. The term 'trusted' here has not been rigorously or consistently defined by all participants in this discussion, but it has taken on meanings ranging from 'correct' to 'of high integrity' to 'free of unspecified side-effects or Trojan horses' to 'being structured and validated consistent with other requirements for elements of the TCB suitable to the TCSEC evaluation class in question'.

It is evident that such discussions are especially relevant if the view definition is used for the dynamic determination and enforcement of view classification properties (MAC) [DEN87]. Most of the controversy, however, revolves around the question of such view definitions in discretionary access control (DAC) contexts.

Verification of compilers and their semantics is known to be a difficult problem. Recent work by Crocker, Landauer and Redmond [CLBR89] has suggested that it may be significantly easier and more fruitful to compare formally the actual executable object code images with source code (or its program specification) than to attempt compiler verification. This has led us to suggest that, for the interim, it may be adequate to configure a well-known, tested compiler whose immutability is assured by an OS TCB into a TDBMS application context such that every input and every output of the compiler can be captured and recorded by a mechanism of sufficiently high assurance to satisfy the ancient audit requirement: To provide the witnesses and vouchers required to support an "official examination of accounts ... so as to ascertain their accuracy."

[17] Again, it is unclear that one can conclude that the selected tuples were modified as a consequence of the query being processed at the time, or as a consequence of some previous event that may have been queued (legitimately or not) for later processing.

Chapter 8

Conclusions and Recommendations

This paper has explored several issues relating to audit in its historical and trusted systems contexts. It has been found that little guidance is provided for either what needs to be audited in TDBMS contexts or *when*, *how*, or even *where* audit data should be captured or recorded. We have also found that surprisingly little attention has been given in practice to the analysis of TDBMS audit data. It appears to be fruitful to explore the use and correlation of data collected from several different layers of combined systems architectures.

There also appears to be great utility in pursuing the analysis of an audit trail's potential credibility as a means of assessing the assurance and trust characteristics of system security architectures. Tentative conclusions drawn from preliminary application of this analysis technique have been presented. Research is continuing in this area.

Appendix A: Questionnaires

This appendix contains the final version of the state-of-the-art product and security officer questionnaires. Both these questionnaires were continually evolved during the course of the study.

Appendix A

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable ? Which of these are selectively auditable ?

2. For each of the above, what information can be captured about the event ?

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log ? Is the log single level or multi-level ?

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal ? Between the DBMS audit log and the O.S. audit log ? Are each of these separate ? Can they be correlated easily ?

5. What automated tools are provided for analysis of the audit log ? What additional tools would be useful ?

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools ?

7. What reports and technical papers have you encountered that discuss these questions ? (This should have been the last question, but was left as question 7 for consistency with earlier versions of the questionnaire.)

8. What constitutes a failed access attempt ? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject ? How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool ?

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC) ? How might these be represented in the audit log ?

- Use of locks ?
- Delete down operations ?
- Inference and aggregation attacks ?
- Probing by means of integrity constraints ?

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway ?

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured ?

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

13. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both implicit and explicit, auditable?

14. At what stages in the processing of a query (parsing, compilation, optimization, execution, etc.) can audit capture occur? To what extent is the choice of capture points constrained by the DBMS architecture? How does the choice of capture points affect the credibility, and usefulness of the audit trail ?

15. Other interesting issues, ideas, suggestions ?

Appendix A

Security Officer Questionnaire

The objective of this questionnaire is to gather information about current usefulness of audit trail.

1. What audit information do you typically collect? What type of system (i.e., operating system, database....)?

2. How is this information analyzed and for what purpose? Are tools used? If so, how useful are the tools? What kinds of tool would be useful?

3. Is the information gathered adequate to meet the needs? If not, what additional information would be useful?

4. Why do you collect audit data? Is auditing useful or necessary?

5. How often, or under what circumstances is the audit trail reviewed?

6. How long are the audit logs kept? How are the audit logs kept (file or paper?)?

Appendix B:
State-of-the-Art Survey Data

This appendix contains the questionnaire responses, interview summaries, and system analysis summaries collected during the course of this study. Each organization for which a write-up or questionnaire was completed, is represented in this appendix. The summaries are listed in alphabetical order by organization name.

Atlantic Research Corporation

Product: TRUDATA

Date of meeting: June 16, 1989

Point of Contract: Ronald Knode
Cornelius Haley

Questionnaire Author: Trusted Information System, Inc.

System Summary Author: Trusted Information System, Inc.

Reference:

Knode, R.B., and R.A. Hunt, "Making Databases Secure with TRUDATA Technology," *Proceedings of the Fourth Aerospace Computer Security Application Conference*, Orlando, Florida, December 1988.

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

TRUDATA always audits: attempts to execute commands that TRUDATA does not support that the isolated database machine did support (e.g., stored procedures), integrity lock failures and the process of initialization of auditing.

The selectively auditable events are abort transaction, DAC modifications, back end audit, begin transaction command, tuple insert summary, CCR create/change, reconfigure back end, database functions (open, define, create, drop, extend, dump, load, roll forward, undefined, lock, unlock), set date backend, tuple delete summary, DSL create/change, end transaction command, create/delete index functions, security label change, OAL modification (OAL has privilege to all the database in the system), individual tuple insert, individual tuple delete, individual tuple replace, retrieve and copyout tuples, table functions (associate, define, create, extend, lock, unlock, truncate), set time command, tuple replace summary, view functions (define, create, undefined, lock, unlock).

The text of the query is not captured and neither are selects at the record level.

2. For each of the above, what information can be captured about the event?

abort transaction: flag indicating success or failure (this command in reality should never fail).
DAC modifications: user, entity, old set up, new set up, flag, and error code.
back end audit: object, flag, error code.
begin transaction command: flag.
CRR create/change: entity, low, high, flag, code.
reconfigure backend: flag.
database functions: database, who's function, flag, error code.
set date: date, flag, code.
tuple delete function: table/view, how many tuples came in to TRUDATA, how many tuples went out of TRUDATA, high water security level, flag, code.
DSL create/change: entity, dsl, flag, code.
end transaction command: flag.
index function: object, function, flag, code.
initialize auditing: flag.
security label change: object, bin, oldlabel, newlabel, flag, code.
integrity lock failure: view, bin.
OAL modification: user, old set up, new set up, flag, code.
individual tuple insert: table/view, class, bin, flag, code.
individual tuple delete: table/view, class, bin, flag, code.
individual tuple replace: table/view, class, bin, flag, code.
retrieve, copyout tuples: view, how many tuples came in to TRUDATA, how many tuples went out of TRUDATA machine, high water security level, flag, code.
table functions: table/view, function, flag, code.
set time command: time, flag, code.
unsupported command: command

tuple replace summary: table/view, how many tuples came in to TRUDATA, how many tuples went out of TRUDATA, high water security level, flag, code.
view functions: view, function, flag, code.
tuple insert summary: view, how many tuples the user attempted to created, the number actually created, high water security level, flag, code.

Along with the above information TRUDATA inserts the user's login name from the password file and System V/MLS inserts the date, time, and user's process id. The audit log is stored in ASCII format, because of System V/MLS.

Some audit routines include additional information describing the successfulness of an event. The following describes the varying levels of successfulness of an event.

a. Successful/full: The isolated database machine returned data, and all of the data passed security filtering.
b. Successful/partial: The isolated database machine returned data, but some of the data did not pass security filtering.
c. Successful/noaccess: The isolated database machine returned data, but none of the data passed security filtering.
d. Successful/none: No data was passed back from the isolated database machine.
e. Unsuccessful: TRUDATA did not send the command to the isolated database machine, because the user was not authorized to retrieve the specified data.

3. What special privileges are required for access to the audit control mechanism and the audit log? Is there a need for selective access based on information sensitivities and administrator clearances?

Since TRUDATA is using System V MLS, which has only two privilege sets (i.e., superuser and user), superuser privilege is required to access the audit control mechanism and the audit log. The audit log is owned by root (i.e., only root can read or write to the audit log) and stored at system low.

4a. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal?

Totally separate, the isolated database machine handles the rollback-recovery journal and the operator system handles the audit log.

4b. Between the DBMS audit log and the OS audit log?

TRUDATA writes the DBMS audit records into the audit log kept by the operating system.

4c. Can they be correlated easily?

No work has been done to try to correlate the audit log with the rollback-recovery journal. It appears that this may be useful.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

AT&T includes a primitive audit browsing tool with System V/MLS, but sells an additional audit tool. This additional tool is an untrusted relational database.

A tool to correlate the audit log and the rollback-recovery journal might be useful.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or offline audit analysis tools?

No information.

7. What reports and technical papers have you encountered that discuss these questions?

None.

8. What constitutes a failed access attempt?

TRUDATA auditing mechanism can detect that the user did not get all the information (i.e., the user only got the information that the user had access to receive) or that the user tried to get something that the user was not authorized to access. See earlier description in question #2.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

Since TRUDATA is only a B1 system, they did not explicitly worry about covert channels.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

No.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

The minimum set is the process to initialize auditing, integrity lock failure, and the attempted use of unsupported commands. The documentation may (not yet written) contain a recommended guideline for set the audit flags (i.e., insert, replace, delete plus retrieval summaries).

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

No difference. But, pre-compiled queries is one of the commands that TRUDATA does not support that the isolated database machine does support.

13. Can MAC and DAC constraints be associated with metadata? Are metadata accesses auditable?

Metadata is auditable when a user explicitly accesses the table where it is stored or the trusted Front End's schema. But metadata is not auditable in terms of implicit access to the table. If a user has read access to any part of the database, then the user has read access to all associated metadata.

System Summary: TRUDATA

I. Status

A commercially available product.

II. Architecture

TRUDATA product is based on the Integrity Lock approach. This approach allows a trusted filter to use an untrusted DBMS. The filter mediates all access between users and the DBMS. The trusted filter preprocesses all data inputs to the database by determining the data's security level and computes an unforgeable authenticator. The data to be stored, the security label and the authenticator are given to the DBMS. On retrieval, the filter recomputes the authenticator and compares it with the authenticator returned by the DBMS. If they match, the data's integrity and its associated security level are confirmed.

TRUDATA's architecture is comprised of a trusted operating system, currently targeted for at least the B1 level, to enforce the separation between the DBMS and user. All other trust for data and process security is in the filter (i.e., the trusted front-end).

III. Security Policy

The subjects are users, untrusted applications, and trusted applications (where "applications" are processes acting on behalf of a user). Subjects are "labeled" with an access range in which the subjects can operate. At any point in time, a given subject is operating at a particular access level.

The named objects are databases, relations, and views (pviews and mviews) and storage objects are pview instances. Projections are constrained to form the pviews (primitive views). Mviews (multi-views) are combined (joined) pviews. Pview instances are the intersection between a defined pview and a record (tuple). The relationships between objects are databases contain relations and mviews, which contain pviews, which contain pview instances.

There are three types of object labels. One type, a container clearance requirement (CCR) label, is applied to named objects. It is a classification vector which establishes a floor and a ceiling on the security label of any item in the container. Default security level (DSL) is another type of label which is attached to every named object. It serves to provide the label for all storage objects created whenever an Actual Security Level (ASL) label is not supplied. ASL label is that label stored with each pview instance (i.e., with each storage object).

Discretionary Access Control policies are enforced via the system level Operator Authorization List (OAL), database (relations and views) level Access Control Lists (ACL) and complementary Exception List (EXL). There is only one OAL, which grants operator privileges to subjects at the system level. Each named object may have an ACL and/or EXL. Each entry on an OAL/ACL consists of the subject (or group) identifier and the authorized access modes. EXLs list the subject (or group) and all specifically denied access authorizations (at the given level). Privileges granted at one level are "inherited" by the lower levels. These access checks are performed in the trusted front-end.

Mandatory Access Control is enforced based on the subject and object labels and five subpolicies (create, read, write, update, and remove). These access checks are performed in the trusted front-end.

The current release of TRUDATA provides distinct encryption keys for each database, thus preventing the untrusted DBMS from intermixing data from different database (i.e., circumventing DAC). Future releases will permit distinct encryption keys for each table (and thus for mviews also). Since each database (and later each table) has its own encryption key, a database, when it is created, is empty. Subsequently, all data retrieved from a database must have at some point been inserted into that particular database. With an integrity lock architecture the untrusted DBMS is responsible for deleting tuples. Thus, the filter assures that a tuple is not inappropriately released, not that a tuple is destroyed. With this in mind, a container "theoretically" has the same properties as a laser disk storage device. That is, once data is written to a container, that data cannot be removed.

IV. Auditing

TRUDATA identified categories of database events, that are distinct from operating system events, that are required to be auditable. Briefly, these categories include:

1) successful/unsuccessful database OPEN;

2) a summary of a successful retrieval of any set of objects from a TRUDATA protected database (where retrieval "success" is defined to be any SELECT which retrieves at least one object from the database).

3) unsuccessful retrievals;

4) successful/unsuccessful attempts at creating new storage objects (pview instances);

5) successful/unsuccessful attempts to update (existing) storage objects;

6) successful/unsuccessful attempts to remove storage objects;

7) successful/unsuccessful attempts to create named objects;

8) successful/unsuccessful attempts to remove named objects;

9) Integrity lock failure; and

10) BEGIN, END, and ABORT transaction requests.

11) Attempts to perform unsupported commands.

For more detail on auditing see the TRUDATA questionnaire write up.

V. Other Topics of Interest

1. Failed Access:

In TRUDATA a failed access is any command that was not sent to the back-end DBMS, because the user was not authorized to retrieve the specified data (e.g., typos will be auditable as a failed access).

TRUDATA also has four levels of success, three of these may be considered failures. Successful/partial is when the back-end database returned data in which some of the data passed security filtering. Successful/noaccess is when the back-end returned data in which none of the

data passed security filtering. Successful/none is when no data was passed back from the back-end.

2. Transaction Management:

In TRUDATA, if an auditable event occurs within a transaction, the events are written. But begin transaction, end transaction and abort transaction are also auditable events. The information captured is a flag indicating success or failure of the command. So it is possible to determine from the audit trail whether or not the transaction was committed.

Note the above events must be selected to be audited.

3. Auditing at Different Levels of Abstraction:

Since the DBMS is untrusted, TRUDATA captures all of its audit information after the back-end DBMS executes.

Note: Attempts to execute stored procedures are rejected by TRUDATA.

4. Delayed Binding:

All DAC bindings are recomputed every time a user opens a database. So access to an object remains unchanged until the database is re-opened. The MAC label of the user is obtained once upon TRUDATA start-up. The MAC label of a tuple is checked each time the user accesses the tuple.

5. Control of Audit:

To access the audit trail or to set or change the auditing defaults, the user must have superuser privileges.

Bank of California

Product: N/A

Date of Meeting: July 11, 1989

Point of Contact: Leslie Chalmers

Questionnaire Author: Trusted Information System, Inc.

System Summary Author: N/A

Reference: None

Appendix B Bank of CA

Security Officer Questionnaire

The objective of this questionnaire is to gather information about current usefulness of audit trails.

1. What information does the Security Officer receive in an audit trail? What type of system (i.e., operating system, database...)?

   The bank uses 8 or 9 different application programs running on an operating system with a RACF subsystem. DBMSs make up one-third of these applications. The system of record is the mainframe which feeds information to the DBMS tools.. Currently, audit information is produced only by RACF. When a file is opened RACF audits what user ID opened it and how long the file remains opened.

2. How is this information used? Paper audit trails are reviewed weekly.

3. Is the information gathered adequate to meet the required needs? If not, what information would be adequate?

   No. The information is incomplete and lacking in the granularity of detail. Some information that would be useful is the date, time, user ID, application software used, data accessed, before and after pictures (any modifications). Integrity is the highest priority.

4. Are tools used in reviewing the audit trail? If so, how useful are the tools? What kind of tool would be useful?

   None are used.

5. What is the purpose for auditing? Is auditing useful or necessary?

   The main purpose behind auditing is to determine what authorized people are doing--whether they are doing something improper or not. They mostly care about whether authorized people are abusing their authority. Auditing could be more useful if the audit trail was more informative.

6. When is the audit trail reviewed? Weekly.

   Leslie suggested that we speak with Steve Weiland of the EDP Audit Association in Carroll Streams, IL (near Chicago).

Britton Lee, Inc.

Product: ShareBase

Date of Meeting: July 11, 1989

Point of Contract: Tod Sambar
Michael Ubell
Paula Hawthorne (previously with Britton Lee)

Questionnaire Author: Trusted Information System, Inc.

System Summary Author: N/A

References:

1. *ShareBase II Technical Overview*, Britton Lee, Inc., 1989.

2. System Administrator's Manual, Britton Lee, Inc., 1989.

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

   Sharebase selectively logs updates, deletions, and insertions to any tables. The DBA can turn on logging for tables which were originally created without logging.

2. For each of the above, what information can be captured about the event?

   The following information is recorded:

   - the type of update performed
   - the date and time of the update
   - the user performing the update
   - the table that was updated
   - data that was changed
   - copy of record before

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log? Is the log single level or multi-level?

   The DBA only has access to read (i.e., can't modify), but can grant read access to other users.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

   No relationship.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

   ShareBase provides an audit function which retrieves information in relational form. The audit query can search for a specific activity and determine when it was done and by which user.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

   The use of intrusion detection has not been examined because the customers have no way to request such techniques.

7. What reports and technical papers have you encountered that discuss these questions? (This should have been the last question, but was left as question 7 for consistency with earlier versions of the questionnaire.)

   None.

8. What constitutes a failed access attempt? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject? How might such attempts be represented in te audit log such that they could be detected by an off-line analysis tool?

   Not relevant, because ShareBase does not enforce any access controls.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

   - Use of locks?
   - Delete down operations?
   - Inference and aggregation attacks?
   - Probing by means of integrity constraints?

   Not Applicable.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

    No.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

    There are *no* defaults.

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

    No differences.

13. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both implicit and explicit, auditable?

    Not relevant.

Other Notes and Questions

Sharebase is an untrusted DBMS, but is used as the back-end machine in one of TRUDATA's configurations.

DBA may need to know when changes have been made to data or to database structures, permissions, etc., and by whom. The audit function retrieves information in relational form. The audit query can search for specific activity (for example, deleting a table) and determine when it was done and by which user.

Digital Equipment Corporation

Product: N/A

Date of Meeting: August 10, 1989

Point of Contact: Jay Davison
Kevin Duffy

Meeting Notes Author: Trusted Information System, Inc.

Questionnaire Author: N/A

System Summary Author: N/A

Reference: None

NOTE: Since the researchers at DEC were not currently investigating auditing in database systems, the meeting described by this write-up took the form of an informal discussion rather than an interview structured to follow the survey questionnaire. As a result, this write-up consists of informal meeting notes, and does not follow the format used for the other write-ups.

Summary

Many DEC customers seem more concerned about being able to audit "who changed what" than "who saw what." These are fundamentally different audit objectives that lead to different kinds of audit features.

If the audit data is collected at a level of abstraction that is "too low," for example, auditing the row IDs of every record retrieved from the disk, the audit trail will not only be too voluminous to be useful, it will be misleading; that is, it will reflect the retrieval of much information that was subsequently logically disqualified, and hence never returned to the user. On the relationship between the DBMS security audit log and the OS audit log:

> An advantage to separating the two is that DBMS users cannot deny service to other O.S. users via contention for audit resources or storage exhaustion.
>
> If the two logs are separate, it may be necessary for the O.S. to provide user ID and session ID information to the DBMS so that the logs can be correlated.

In a secrecy-oriented system, there seems to be a need to audit uncommitted transactions. Otherwise a user could start a transaction, fish around for data, and then abort the transaction without leaving a trace. However, auditing all uncommitted transactions may load the audit trail with useless information that may obscure the effective sequence of "interesting" security-related auditable events.

Depending on how audit information is buffered and recorded, and how mutually exclusive access to the audit information is provided, there is the potential that an auditor browsing the audit log in real time may have a negative performance impact (may interfere with) on DBMS users.

George Mason University

Product: N/A

Date of Meeting: June 19, 1989

Point of Contact: Dr. Sushil Jajodia

Questionnaire Author: Dr. Sushil Jajodia

System Summary Author: N/A

Reference: None

Appendix B George Mason U.

DBMS Audit Questionnaire

1. What DBMS events are auditable? Which of these are selectively auditable?

   I believe "zero-information loss" is possible. That is, you can audit every event in your system. What event you actually audit depends on the sensitivity of your data in the system.

2. For each of the above, what information can be captured about the event? At the application level? At the DBMS level?

3. What special privileges are required for access to the audit control mechanism and the audit log? Is there a need for selective access based on information sensitivities and administrator clearances?

   I consider audit records to be most sensitive, from a privacy as well as a security viewpoint. Thus, there is a need for both and MAC and DAC on them.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

   I will address only the database part of this question. I believe that if audit logs are carefully kept (meaning we keep an audit of those events which are relevant for DBMS rollback/recovery), then we can certainly do away with the rollback-recovery journal. It should not be too difficult to correlate them.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

   I believe that audit information should not have an ad hoc structure. We should impose a logical structure on the audit information. Once this is done, we can use a query language, such as SQL, to query the audit information.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

   I think audit is the basis on which we can build automated intrusion detection tools. Audit information augmented with some "knowledge" is the key to building such tools. Also, audit information can be used to limit inferences.

Harris

Product: N/A

Date of meeting: N/A

Point of Contract: Rhonda Henning

Questionnaire Author: Harris

System Summary: N/A

Reference: N/A

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

Auditable events should include:

creation of a database
creation of additional tables
creation of additional views
database recovery/restructuring tools
access control operations
user sensitivity range definitions (if different from OS)
data creation, modification, deletion, selection
importation of data in bulk
exportation of data in bulk
turning on/off the audit log
deletion of audit information
deletion of recovery information
reading of audit data

All should be selectively auditable for a given user with selection specified by the database security administrator. The database security administrator has the capability to determine which events should/shouldn't be audited with regard to a given database.

2. For each of the above, what information can be captured about the event?

creation of a database /date/time/userid/groupid/success/failure cause/from schema file/

creation of additional tables /date/time/userid/groupid /success/failure cause/to database/table name/table attributes/table items/

creation of additional views /date/time/userid/groupid/success/failure cause/to database/view name/view attributes/view items/view level range/

database recovery/restructuring tools /date/time/userid/groupid /operation/success/failure/which database/number of records/

access control operations /date/time/userid/groupid/operation/table/view/data item/database/receivinguser/success/failure/

user sensitivity range definitions (if different from OS) /date/time/userid/groupid/operation/table/view/data item/databases/definition/receiving user/success/failure cause/

data creation, modification, deletion, selection /date/time/userid/groupid/operation type/query/success/failure cause/

importation of data in bulk /date/time/userid/groupid/filename/recieving database/receiving table/success/failure cause/

exportation of data in bulk /date/time/userid/groupid/filename
written/from database/query/success/failure cause/turning on/off the audit log
/date/time/userid/groupid/audit/filename/action/success/failure cause/

deletion of audit information /date/time/userid/groupid/audit
filename/success/failure cause/

deletion of recovery information /date/time/userid/groupid/recovery
filename/success/failure cause/

reading of audit data - /date/time/userid/audit filename/ success/failure

3. What special privileges are required for access to the audit control mechanism and the audit log? Is there a need for selective access based on information sensitivities and administrator clearances?

The audit control mechanism and audit log are database administrator privileges or database security officer privileges. Access to these mechanisms is controlled through the operating system's MAC and DAC mechanisms.

By virtue of the fact that users granted these roles are considered trusted subjects, no additional selective access should be required for them.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

   a. The DBMS audit log should be distinct form the rollback/ recovery journal. The audit log should contain information considered "security relevant" to the operation of a secure DMBS. Therefore, the DBMS security functions should not care about collecting raw DBMS data on its way to an end-user. However, the queries made by a user are auditable information that needs to be stored in the audit log. If necessary, rollback/recovery journals can be used to restore a database based on audit log query captures.

   b. The DBMS audit log provides a finer granularity of control and definition than the operating system audit log. The underlying operating system is responsible for audit of file manipulation commands which may be executed on the behalf of the database management system or a single user. The DBMS is responsible for auditing activity beyond basic file system audit capabilities.

   c. Yes, all are separate, but each is necessary. Correlation depends on the compatibility between the three audit logs. If all are in a similar format, correlation should be straight-forward. If formats differ, correlation becomes a more difficult problem. By similar format, it is a given that the logs do not all contain the same information. However, there should be enough similar information in each audit log to allow relatively easy correspondence. We assume, of course, that this is done with the aid of audit reduction tools.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

Most DBMS provide no additional tools beyond the standard query language and pattern matching techniques found in basic text editors. If a pattern for potential intrusion can be

expressed as an SQL-type query, the audit log can be analyzed for it. Otherwise, it is basic by-hand analysis.

Two types of tools: one similar to existing intruder detection systems, only with scenarios defined for DBMS, that could determine if the system appeared to be functioning normally and flag "abnormal" behavior for further analysis. Another, that would identify the DBA once a definitive pattern of suspected compromise has been indicated and notify him, ostensibly while a hostile user was still logged in.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

See 5. While off-line audit analysis tools are currently acceptable, the delay involved in detection of possible penetration makes it imperative that more timely techniques be developed.

7. What reports and technical papers have you encountered that discuss these questions?

Strictly from memory, there haven't been very many (and, at the time, none come to mind) that specifically address auditing in a DBMS from a security perspective. LDV and SeaViews and perhaps the E-R model all addressed it from a theoretical perspective. Most of the database engineering conferences have rollback/recovery papers included.

The MCC work on ORION had an access control policy, but offhand, I don't remember if a specific audit was included.

8. Other ideas, issues, suggestions?

Probably need to address:

system performance with various auditing granularities
whether or not storing the actual output from reporting queries is useful
whether or not audit logs per security level make sense
if OS intruder detection techniques are appropriate in a DBMS

Honeywell

Product: LOCK Data Views

Date of Meeting: June 21, 1989

Point of Contact: Paul Stachour
Dick O'Brien

Questionnaire Author: Honeywell

System Summary Author: Trusted Information System, Inc.

Reference:

Dwyer, P., E. Onuegbe, P. Stachour, and B. Thuraisingham, "Query Processing in LDV: A Secure Database System," *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, April 1988.

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

This note responds to the RADC-sponsored DBMS Audit Questionnaire of May 15, 1989. It gives the LOCK Data Views thoughts as of July 11, 1989. This note should be interpreted as the LDV team's suggestions, and not as a definitive design, or build style.

1. What DBMS events are auditable? Which of these are selectively auditable?

#1. Change to a value of a tuple of a relation.
#2. Insertion of a new tuple.
#3. Retrieval of a tuple during a response.
#4. Creation of a new metadata item, such as a table, attribute, or constraint.
#5. Update of an item of metadata.

Metadata items (#4,#5) are always audited. Data items (#1,#2,#3) may be selective. The selection criteria on the data is not yet defined.

Note that due to the LDV multi-level design of the metadata, an action such as an attempted retrieval of the values of an attribute where the existence of the attribute is classified above the level of the requesting process is not an auditable event, since the portion of the DBMS validating the attributes cannot even determine the attribute existence itself. Similarly, due to the LDV data storage mechanism, attempts to fetch data which is classified beyond the level of the process is not an auditable event either.

2. For each of the above, what information can be captured about the event?

In LDV, the DBMS does all the audit-logging of DBMS audit events. [LOCK also does audit-logging of OS-level audit information, but that is outside the scope of this questionnaire.] In general, it is not able to capture application-level data due to the domain/type mechanism of LOCK. The DBMS is isolated from the application, and does not have access to application-information, such as the customer the user-program is working with, etc. It can, and does, capture OS-level information associated with the logged-in user, such as identification of user, security-level of request, current-time, etc., since this data is available to any LOCK subject.

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log? Is the log single level or multi-level?

The LOCK domain/type mechanism is used to separate the audit-log from the users. For each database, internally there is a set-of-files (up to 1 file per each security level within the database) of a particular type that is writable by DBMS code only, and readable only by DBSOs. In addition, the normal LOCK DAC applies, so that one DBSO will not be able to read the audit-log of a different database. The system security officer(s) would have read-permission on all the DBMS audit-logs. The structure of the audit-log mirrors that of the database, including discretionary and elective permissions. We visualize the audit-log as very similar to "just another relation in the metadata" with a special-name (for example, DB$AUDIT_LOG).

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the O.S. audit log? Are each of these separate? Can they be correlated easily?

No relation between DBMS audit log and DBMS rollback-recovery journal. We believe that they have different visibility and purge-time needs. Merging the two would be more work that worth. No relation between DBMS audit log and OS audit log. The OS audit log could potentially be over-run with application data, thus forcing a log-swap and corresponding system-delay and thus a covert channel as well as denial-of-service when needed. We believe that an OS-provided application-audit-log facility is needed and useful in the general case. Because of the LOCK domain/type mechanism, we can "get away" with having the application roll-their-own for our exploratory design/implementation.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

The normal DBMS relational ability to extract a particular subset of events, followed by a pattern-search outside of the DBMS, if necessary.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

Within the DBMS, there would be an audit-log manager through which all DBMS-style audit-data would pass. One could specify to that audit-log manager at what points or what threshold should result in equivalence of message to a system security officer or something placed on the system audit log.

Questions add by Paul:

7. Please give your name. Are you willing/able to share database and/or implementation strategies with us? Could experiments be conducted on your system/prototype?

This work is being done as part of the LDV Project at SCTC. All the LDV work is shareable when written as the final-deliverable version. Preliminary work is often shareable at draft-status upon request to our contracting officer. We believe that experiments on our exploratory development model, when built, would be approved by the contracting officer of our customer. However, any such experiments, and time involved in conducting such experiments, would need to be cleared through them.

8. Any additional comments?

Audit MUST be done in such a manner that it is absolutely clear WHO is doing the event, under WHAT circumstances they are performing it, WHEN it is being done, WHERE the (SQL-style) relationships are, and HOW (like what initial comment) the need to provide the audit-data arose.

The DBMS must be built such that the DBSO/SSO can know that these are correct. This means that in the absence of a mechanism such as the LOCK domain/type structure, some other assurance MUST be present to ensure that the audit-data is properly gotten and recorded.

System Summary: LOCK Data Views

I. Status

Research Prototype.

II. Architecture

Lock Data Views (LDV) is a multilevel secure DBMS hosted on the LOCK trusted computing base (targeted beyond A1).

III. Security Policy

The LDV security policy builds on the concepts of the LOCK security policy (i.e., satisfies the A1 level) and extends it through the incorporation of an explicit classification policy. The classification policy must address those factors which are crucial to a correct determination of the sensitivity level of data within the DBMS context. LDV policy considers:

Name-dependent classification: rules that refer to data items by name. This provides classification at the granularity of relations and attributes.

Content-dependent classification: rules that refer to the content of data item occurrences. This provides classification at the granularity of tuples and elements.

Context-dependent classification: rules that refer to combinations of data items. This can be used to reflect sensitivity of specific fields when accessed together.

Inference control: the determination of data sensitivity based on the potential inferences that can be made based on a sequence of access requests.

Both MAC and DAC security policy are enforced by LOCK. Accesses to data as well as metadata are controlled by LOCK. Information in the database and metadata are stored in single level files (i.e., LOCK objects). LOCK ensures that these database files may be opened for read/write operations only by subjects executing at the appropriate levels and in the appropriate database domains.

IV. Auditing

In LDV, the DBMS does all the audit-logging of the DBMS audit events. Such as:

1. Change to a value of a tuple of a relation.
2. Insertion of a new tuple.
3. Retrieval of a tuple during a response.
4. Creation of a new metadata item (table, attribute, or constraint).
5. Update of an item of metadata.

LOCK does audit-logging of OS-level audit information (i.e., captures domain/type mechanism enforcement by LOCK).

V. Other Topics of Interest

Inference

LDV maintains a set of history files that are built as responses are released and that are consulted prior to the release of any response. Whenever data is released from the database, a record containing the following information is generated:

1. The user whose subject posed the query
2. The level of the subject that posed the query
3. The set of files consulted in building the response
4. The date-time at which the response was built

These records are stored in objects of type history file, and objects of this type can only be modified by subject in the history maintenance domains. Before a response is generated, a standard database procedure for mapping relations and attributes to files is invoked to determine which files should be consulted during the query processing. This set of files is processed against the history files using the information found in the constraint files maintained by the database security officer, and those files in conflict are removed from the set. This ensures that data which is high level with respect to the constraints and history will not be used to generate low-level responses. The query processing then proceeds in a manner similar to that used in a conventional DBMS systems to retrieve tuples from the files in the set.

International Business Machines, Inc

Product: DB2 Version 2 Release 1

Date of Meeting: July 14, 1989

Point of Contact: Roger Miller

Questionnaire Author: IBM

System Summary Author: Trusted Information System, Inc.

References:

1. DB2 Version Release 1 Audit Trace Usage Guide, IBM International Technical Support Center, Santa Theresa, CA, December 1988.

2. Section 4. Security and Auditing of the DB2 System and Database Administration Guide.

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

The following is a list of all auditable events:

- denied access attempts
- results of GRANT and REVOKE statements
- results of CREATE, ALTER, and DROP operations affecting audited tables
- changes to audited tables
- read accesses to tables identified for AUDIT ALL
- binding of static and dynamic SQL statements
- assignment or change of authorization ids, and utility phases

The audit events can be selected by class of events and can also have additional selection criteria. Tables can have their initial access audited (AUDIT ALL), any changes but not read access audited (AUDIT CHANGES), or no auditing (AUDIT NONE).

2. For each of the above, what information can be captured about the event?

See references.

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log? Is the log single level or multi-level?

The audit mechanism is controlled by the TRACE privilege. Traces can be sent to SMF or GTF and access to this data is generally secured by the security subsystem, such as RACF. Online monitoring of the audit trace can be performed by a person with the MONITOR2 privilege. The AUDIT attribute for a table can be set by the table owner, someone who has ALTER authority for the table; someone who has DBADM authority for the database; or someone with SYSADM authority. DB2 only supports single level security at this point.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

The DB2 recovery log is separate from the audit log, but data from them can be correlated using information in the data headers. Generally, the DBMS is kept with other OS data, in SMF. DB2 data can also be written to GTF, and correlation is relatively easy.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

The audit log can be printed with many variations by the DB2 Performance Monitor (DB2PM). Several other vendors provide tools for this analysis as well. DB2PM can provide the data in a form to be loaded into tables so that SQL and reporting tools like the Query Management Facility (QMF) can be used for analysis. Other analysis products, such as SAS, can also be used.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

Automated intrusion detection techniques can use an online interface to the audit data called the Instrumentation Facility Interface. They can also process the SMF data directly or use SQL to process the data after it is loaded into tables.

7. What reports and technical papers have you encountered that discuss these questions?

The primary discussion of DB2 auditing is in DB2 manuals. The DB2 System and Database Administration Guide (SC26-4374) includes a section on Security and Auditing. A separate manual, the Audit Trace Usage Guide (GG24-3300) was developed by the International Technical Systems Center to discuss usage considerations. There have been several presentations at user groups and security conferences discussing the function and its usage.

8. What constitutes a failed access attempt? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject? How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool?

A failed access attempt is any request for which the current authid does not have adequate authority. The DBMS includes a specific class for such failures and identifies them with an IFCID 140 so that processing these records is simple. DB2PM includes a specific report for detected access failures.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

- Use of locks?
- Delete down operations?
- Inference and aggregation attacks?
- Probing by means of integrity constraints?

The use of locks is implicit in DB2. Locks can be audited, but providing that amount of information causes a significant performance impact. DB2 can include very extensive information in the audit data, but the more typical usage is for performance analysis. The SQL usage can be analyzed, providing an indication of the access intent. Inference and aggregation can be found by small variances in the SQL statements, but this is also typical of statistical analysis of data.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

The Instrumentation Facility Interface (IFI) can be used to detect various failures and then notify a security officer.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

For a security-conscious site, the recommendation is to collect all audit data. Utility data is commonly excluded. The accounting and statistics data should also be kept.

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

The audit data captured from an interactive program can be the same as that for an applications program, if desired. The SQL statements being passed from the interactive program are usually included in audit data. A program that has been precompiled has stored the SQL and it can be traced only when the program is bound or as the program is executed too (using a performance class trace).

13. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both implicit and explicit, auditable?

DB2 does not yet include MAC constraints. DAC constraints can be associated with metadata access. Not all metadata accesses are auditable. THe DB2 catalog contains most metadata and can not be audited.

System Summary: DB2

I. Status

Product is commercially available.

II. Architecture

DB2 is a relational database, providing single-level security that runs as an application on an operating system. It is recommended that it be used in conjunction with Resource Access Control Facility (RACF), a security subsystem. RACF would provide access control to DB2 as well as to the DB2 data sets. (Data in DB2 is stored in data sets.) IBM also offers as an alternative to RACF, the possibility of using VSAM or MVS passwords to protect the data sets.

III. Security Policy

Data access control within DB2 is broken down into three areas--control of access to DB2, data set protection and access control within DB2. The first two requirements can be addressed by RACF or a secure operating system as mentioned above. DB2 controls access to its objects by a set of privileges, each of which allows an action on some object. The following are DB2 objects:

* Database: set of objects for authorization, operations
* Tablespace: area of database, data sets containing tables
* Table: collection of rows with same columns
* Index: ordered set of pointers to data in a table
* View: alternate representation of data
    * may be subset of table
    * may be from multiple tables
* Plan: representation of SQL statements in program

A subject or process in DB2 is represented as by a primary authorization identifier (ID) and possibly one or more secondary IDs. A primary authorization ID represents the process to DB2 and a secondary authorization ID, which is optional, can hold additional privileges available to the process (i.e., a group id or role). IDs can hold privileges that allow them to take certain actions. There are three ways within DB2 to give an ID access to data--the ID can be explicitly granted a privilege or have a privilege revoked, the ID can own an object by having the privilege to create objects, and the ID can acquire the privilege to execute an application plan from a plan owner.

An application plan can contain SQL statements, performing a variety of actions on many tables, all of them requiring one or more privileges. In each case, it is the owner of the plan that must hold the privilege. For another ID to execute the plan, it must have the EXECUTE privilege on the plan, either granted by the plan owner or by virtue of SYSADM authority, but needs no other privilege. Therefore, an ID with the EXECUTE privilege can exercise all the

owner's privileges that have been used in validating the plan, but it only within the restrictions imposed by the SQL statements in the original plan.

In addition DB2 also defines sets of related privileges, called "administrative authorities". By granting an administrative authority to an ID, you grant all the privilege associated with it in one statement.

Note on composite privileges. A SQL statement can name more than one object and therefore require more than one privilege. Such a statement may be executed dynamically even though not one of your IDs has all the privileges required if the set of your IDs together have all the required privileges. If you embed the same statement in a host program and bind the program, the validation will fail.
As in this case, all the required privileges must be held by the single ID that owns the plan.

Implications of the revoke statement. If a privilege is revoked from the ID, its deletion can cascade to other IDs, with side effects that are not immediately evident. For instance when a privilege is revoked from authorization ID X, it is also revoked from any ID to which X granted it, unless that ID has it also from some other source.

In addition if a privilege is granted on a table, it is not necessarily granted on any views of the table. But if a privilege is revoked from a table, it is revoked from a view of that table. If as a result of that loss, the owner of a view no longer has SELECT privileges on it, then DB2 drops the view.

Along the same line, if the owner of an application plan loses a privilege required by the plan, DB2 invalidates the plan.

IV. Auditing

DB2 provides two types of audit: security audit, used to verify the authorization procedures and data audit, used to verify that access to sensitive data is performed as intended by the authorization procedures. The DB2 audit mechanism creates records of actions of certain types and sends them to a designated destination. The audit mechanism is DB2 audit trace and is controlled by the TRACE privilege.

Data audit monitors access to data stored in DB2 audited tables. The audit attribute for a table can be set by the table owner; someone who has ALTER authority for the table, someone who has DBADM authority for the database; or someone with SYSADM authority. The AUDIT attribute for the table can be (AUDIT ALL) initial access audited, (AUDIT CHANGES) any changes but not read access audited, (AUDIT NONE) or not auditing.

Security audit monitors the authorization procedures implemented for a DB2 site. In contrast to data auditing, security auditing once activated, collects data for the whole subsystem. It is independent of the audit specification for a table.

The actual data changed is not recorded (it is recorded in the log). If an agent or transaction accesses a table more than once in a single unit of recovery, only the first access is recorded in the audit log; and then, only if the audit trace is started for the appropriate class of events.

The list of auditable events follows.

1. Authorization failures or attempted security violations.

2. Results of explicit GRANT and REVOKE statements. (Does not include implicit grants and revokes.)

3. Results of data definition (CREATE, ALTER and DROP) operations affecting audited tables.

4. Changes to audited tables (INSERT, DELETE or UPDATE). Only the first attempt to change a table, within a unit of recovery, is recorded. (If the agent or transaction issues more than one COMMIT statement, there are correspondingly many audit records.) The changed data is not recorded, only the attempt to make a change. If the change is not successful, and is rolled back, the audit record remains; it is not deleted. This class includes access by the LOAD utility.

5. All read accesses (execution of SELECT statements) to tables as AUDIT ALL. As in class 4, only the first access within a unit of recovery is recorded.

6. The bind of static and dynamic SQL statements of the following types:

   INSERT, UPDATE, and DELETE statements for audited tables. Except for the values of host variables, the entire SQL statement is contained in the audit record.

   SELECT statements to tables identified as AUDIT ALL. Except for the values of host variables, the entire SQL statement is contained in the audit record.

7. Assignment or change of an authorization ID, whether through an exit routine (default of user-written) or a SET CURRENT SQLID statement.

8. The start of utility job and the end of each phase of utility.

There are three possible destinations for the audit trace: the System Management Facilities (SMF), the Generalized Trace Facility (GTF), and the Instrumentation Facility Interface (IFI) application program.

Data from SMF and GTF can be reduced and put into readable form by the DB2 Performance Monitor (DB2PM). DB2PM can also create summary audit reports, create detailed audit reports for each audit record type, create detailed trace listings of all audit records, and produce sequential files that are loadable into DB2.

Files generated by DB2PM can then be used by analysis tools such as SQL and Query Management Facility (QMF) to provide additional reporting capabilities. QMF allows audit data to be selected according to various criteria and to create reports that can be used to effectively monitor the security objectives of a DB2 environment.

IFI output is collected by a user written application program.

Online monitoring of the audit trace can be performed by a person with the MONITOR2 privilege.

Other Topics of Interest

Integrity Constraints. DB2 provides many controls that can be applied to data entry and update. Some are automatic, some optional. The assignment of column data types and lengths provides some control on the type of data. Creating unique indexes to a column or set of columns in question can assure unique data values where required. It is also possible to make sure that data has a required pattern or is within a specific range.

DB2 also has a CHECK utility that can check the consistency of indexes and the referential constraints on data files.

Referential Integrity. DB2 automatically enforces the rule that every value of a foreign key in a dependent table must be a value of the primary key of the appropriate parent table.

Concurrency Control. Control is handled by "locks" and is automatic. No program can access data that has been changed by another program but not yet committed. Where an application program writes data to both DB2 and IMS/VS, or DB2 and CICS, the subsystems prevent concurrent use until the program declares a point of consistency.

The MITRE Corporation

Product: N/A

Date of Meeting: August 9, 1989

Point of Contact: Rich Graubart
Marie Collins
Bhavani Thuraisingham

Meeting Notes Author: Trusted Information System, Inc.

Reference: None

NOTE: Since the researchers at MITRE were not currently investigating auditing in database systems, the meeting described by this write-up took the form of an informal discussion rather than an interview structured to follow the survey questionnaire. As a result, this write-up consists of informal meeting notes, and does not follow the format used for the other write-ups.

SYBASE TDBMS Demonstration

MITRE is a beta test site for the B1 version of SYBASE Secure Data Server, a trusted DBMS (TDBMS). During our visit, Marie Collins demonstrated the auditing features of the SYBASE product. The following points were discussed during the demo.

SYBASE has a "relaxation property switch". If this switch is enabled, polyinstantiation is turned off. This allows explicit and implicit (as a result of upgrading) delete-down covert channels. There do not appear to be any special audit collection switches expressly for the purpose of focusing on possible exploitation of these channels. However, it is probably easy to detect such exploitation by querying the audit trail. For example, the auditor could select all modify or delete queries where the user's session level is higher than the classification of the row being modified or deleted.

Also note that polyinstantiation DISGUISES impermissible attempts to write-down, by transforming them into permissible attempts to create new objects. This may serve as a good example of how the definition of a "security-relevant event" is architecture or policy specific. On one system, this would be a failed attempt to violate policy; on another system it's simply a successful attempt to create a new object.

"Keystroke" auditing. Contrary to what we had previously thought, the keystroke audit data is "raw," and does not include the translation of views or stored procedures into references to underlying tables. Therefore, if a malicious user thought the audit analysis was going to be based solely on examination of raw query text, the user might be able to partially disguise his/her efforts by using views with deceptive names.

Round-table Discussion

Potential concerns about trusted DBMSs that rely on untrusted backend machines (distributed architecture).

The OS TCB and Trusted Front End (TFE) on the host can't be sure what files (databases and tables) are being opened on the backend in response to a user's request. This calls into question the credibility of the OS auditing log, which is supposed to be able to record which files were opened/closed for each user. Furthermore, it may be difficult to assure that the data returned to a user was fetched from the correct database or table, and therefore complies with DAC constraints. This may indicate that the architecture is too loosely coupled, i.e., the TCB doesn't know enough about what the backend is doing, and can't exercise any control over its behavior.

A similar issue concerns user requests to delete tuples, tables and databases. These events may be recorded in the TDBMS audit log as having been successful, yet there is no assurance that the untrusted backend has actually deleted these objects. This casts doubt on the credibility of the TDBMS audit log. Also, since the C2 requirements for object reuse (residue erasure) seem to depend on the untrusted backend, can there be any assurance that these requirements are enforced? On the other hand, one could view this as an issue of database integrity rather than confidentiality in the same vein as questioning the assurance that an update query was executed correctly. Integrity requirements of this sort probably require that the whole DBMS be trustworthy.

Regarding the example of retracting an Aids false positive test: if fulfilling this need depends on use of the rollback/recovery log, this places trust requirements on the rollback/recovery log. However, in a backend architecture the rollback/recovery log is untrusted.

Rich Graubart said that the CMW audit log (containing mostly file open/close event records) was rumored to grow at the ballpark rate of one megabyte per day. Rich wasn't clear how many users were active or what kind of workload this was for (development? prototype testing by customer?, etc.)

National Computer Security Center

Product: N/A

Date of Meeting: July 6, 1989

Point of Contact: Eric Shellhouse

Questionnaire Author: Trusted Information System, Inc.

System Summary Author: N/A

Reference: None

Security Officer Questionnaire

The objective of this questionnaire is to gather information about current usefulness of audit trail.

1. What audit information do you typically collect? What type of system (i.e., operating system, database....)?

   DOCKMASTER, a Multics Operating System, audits all successful and failed logouts, I/O activities, modification and denials to files, TCP/IP activity, activities entered on the console, and process modification.

2. How is this information analyzed and for what purpose? Are tools used? If so, how useful are the tools? What kinds of tool would be useful?

   DOCKMASTER's audit data is examined through the use of various application tools. What follows is a brief description of the security tools implemented for DOCKMASTER.

   A security absentee job operates on the audit data each night checking for any changes to critical system files and directories. This report is printed out and reviewed by the system administrators and security officer every morning. Daily certain files and directories are monitored and if any changes occur a console alarm is triggered, a message is typed on the console, and output is queued on the line printer for the security officer. This information is registered real-time to aid the security officer in taking action.

   Login denied reports are generated on a daily basis and are reviewed by the security officer. This report is generated by an absentee that runs each morning.

   A report called the access summary which delineates any bad accesses on the system hierarchy again is generated daily. The security officer reviews these files looking for system critical entries.

   The last and most vial tool for analyzing audit data on DOCKMASTER is known as the Multics Intrusion Detection and Alerting System (MIDAS). It is a real-time expert system designed to increase the security of DOCKMASTER. The system is an audit reduction tool that processes audit information as it occurs, looking for items of security significance. This system acts as a security analysis tool that the CESSO can use to better process the audit information that is generated by DOCKMASTER. The audit data created by DOCKMASTER is transferred to the SYMBOLICS system where the MIDAS application resides. Once a security relevant item is detected, the system highlights the occurrence (by adding it to a daily security report) and attempts to alert the security officer or system or system

administration personnel. It is a valuable tool and contributes significantly to the daily secure operations of DOCKMASTER.

3. Is the information gathered adequate to meet the needs? If not, what additional information would be useful?

The information gathered is adequate, but the analysis process could be more straight forward if the formats of the audit records were consistent and a common denominator existed (i.e., beside data and time).

4. Why do you collect audit data? Is auditing useful or necessary?

Not asked.

5. How often, or under what circumstances, is the audit trail reviewed?

The audit trail is reviewed when MIDAS alerts the security officer of a possible intrusion. Also, each day the Security Officer reviews the previous day's audit log.

Oracle Corporation

Product: ORACLE RDBMS Version 6.0

Date of Meeting: July 14, 1989

Point of Contact: Linda Vetter
William Maimone

Questionnaire Author: Oracle Corporation and
Trusted Information System, Inc.

System Summary Author: Trusted Information System, Inc.

References:

1. ORACLE RDBMS Database Administrator's Guide Version 6.0, ORACLE Corporation, Belmont, CA, April 1989.

2. The ORACLE Secure RDBMS Project, ORACLE DBA Guide V6.0 Addendum, Martin Gruber, ORACLE CORP., May 5, 1989.

DBMS Audit Questionnaire

1. What DBMS events are auditable? Which of these are selectively auditable?

The second half of this question is the easiest to answer. Each individual "auditable event" may be selectively turned on and off by a user with proper privileges.

The following types of events can be audited if selected in Oracle Version 5.1:

Table level operations which can be audited:

ALTER, AUDIT, COMMENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE.

View level operations which can be audited:

AUDIT, COMMENT, DELETE, GRANT, INSERT, LOCK, RENAME, SELECT, UPDATE.

Note: The comment command simply allows a text string to be associated with a table or a column. Its creation can be audited.

The following operations on sequences (sequences are used to create unique keys) can be audited:

ALTER, AUDIT, GRANT, SELECT.

The following system audit options can be set:

DBA	-	audit the execution of all statements requiring DBA role.
CONNECT	-	audit connection to and/or disconnection from the database.
RESOURCE	-	audit statements requiring RESOURCE privilege (e.g., creating objects)
NOT EXISTS	-	audit statements which result in an "object does not exist" error.

For each auditable event, successful and/or unsuccessful attempts to perform the operation may be audited.

For each auditable event, an audit record can be created on every access or once per database session on first occurrence.

Auditing options can by set be default on all tables or all system options.

Database start-up and shutdown will be audited in the OS audit trail.

Deletions from the audit trail will be tightly controlled through OS validation and auditing in the OS audit trail.

Auditing can be set on specific database objects for specific operations. This means that auditing can be set on for a specific table or view for a specific operation on that table or view (i.e., select, update, etc.).

Auditing may also by set on for specific DBMS users. This may be done with the same flexibility of auditing by DBMS object.

2. For each of the above, what information can be captured about the event?

All audit records contain the following information:

sessionid - numeric ID for each database session.
entryid - numeric ID for each audit entry in a session.
statement - numeric ID for each statement run.
timestamp - date and time of the audit entry or logon/logoff.
userid - the user's database ID.
userhost - instance ID for the Oracle instance.
terminal - identifier of the user's terminal.
action - numeric code for the action attempted.
returncode - Oracle return code generated by the action (zero = success).
obj$creator - owner of the database object, if any.
obj$name - name of the database object, if any.
auth$privileges - privileges granted/revoked by a GRANT/REVOKE statement.
auth$grantee - the name of the grantee in a GRANT/REVOKE.
new$name - the new name in a RENAME.
ses$actions - session summary, one character for each type of action.
logoff$lread - number of logical reads during a session (number of pages actually touched).
logoff$pread - number of physical reads during a session (number have pages copied from disk).
logoff$lwrite - number of logical writes during a session.
logoff$dead - deadlocks detected during the session.
logoff$time - date and time of logoff.
comments$text - comments.

Note: Oracle does not record the data which was returned as the result of a query, nor do they save the query.

Note: The logoff$ items could be useful in intrusion detection.

3. What special privileges are required for access to the audit control mechanism and the audit log? Is the log single level or multi-level?

In order to enable auditing on an object, the user must own the object or have AUDIT access on the object. A user who has enabled auditing actions on an object can see all resulting audit entries by querying the view USER_AUDIT_TRAIL.

In order to enable auditing of system actions or to set default auditing options, the user must have AUDIT SYSTEM privilege. The view DBA_AUDIT_TRAIL shows all audit records.

In version 7.0, deletions from the audit trail will be further limited so that a special OS privilege is required to delete any record from the database audit trail with that deletion audited in the OS audit trail.

In a single- or multi-level system, all audit records are stored in a single DBMS table or audit records can be directed to an operating system audit trail. Several views are used to control the access to this table based on privilege and on ownership.

In a single-level system, the audit log would be multi-level as well. Audit records are always stored at the level where the action originates.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

The audit log is completely separate from all journals.

The Oracle DBMS has a separate "redo log" used solely for media recovery. Transaction rollbacks are supported by "rollback segments" which are stored in database files along with user data and metadata. The DBMS audit trail is stored as an ordinary database table.

Note: The DBA guide states that the redo log is "low-level representations of changes that cannot be related to user actions, the contents of a redo log file should never be edited, altered, nor used for any application purposes, such as auditing."

The DBMS audit trail is not directly related to the OS audit trail, though the DBMS may make entries into the OS audit trail. The difficulty of correlating the two is unclear, and probably not well defined given the differences in the granularity of objects and operations. However, it should be possible to partially correlate the OS audit and the DBMS audit. This is because the DBMS uses the OS user id and records the terminal id which is connected to the DBMS. This information can be used to correlated the two logs. How easy this correlation is really depends on the flexibility of the OS audit log. If the OS audit log can be transferred easily to a relational database, the SQL can be used to join the two logs based on user or terminal and time.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

There are currently no automated tools for audit log analysis, but a number of useful database views are defined to aid in audit trail analysis. Full descriptions of these views can be found in the Database Administrator's Guide. The following is a list of these views:

ALL_DEF_AUDIT_OPTIONS	- Default audit options for newly created objects.
AUDIT_ACTIONS	- Description table for audit trail action type codes.
DBA_AUDIT_DBA	- *Audit records of all DBA activity.*
DBA_AUDIT_EXISTS	- Audit records for all NOT EXISTS errors.
DBA_SYS_AUDIT_OPTIONS	- Description of current system auditing options.
DBA_TAB_AUDIT_OPTIONS	- Description of current table auditing options.
USER_AUDIT_CONNECT	- Audit trail records for user logons/logoffs.
USER_AUDIT_RESOURCE	- Audit trail records for use of resource privilege.
USER_AUDIT_TRAIL	- Audit trail records relevant to the user.
USER_TAB_AUDIT_OPTS	- Auditing options for the user's own tables and views.

Since the audit log is stored as a relational table, the full expressive power of the *SQL language and all tools normally provided by Oracle (e.g., Sql*Forms)* can be used. These tools are useful for getting the data together in a form that can be easily analyzed however, an automated analysis tool which could take advantage of the relational format would be useful.

Other useful tools would include:

- A tool which would dynamically monitor and display the contents of the audit trail and take appropriate action if a penetration attempt is detected.

- A tool to produce security reports, including a description of any changes made to the security profile (grants, revokes, new users, etc.)

6. In what way could *automated intrusion detection* techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

These type tools, if able to use the SQL language, could quite easily and usefully be integrated as application packages. They would most likely need to run as privileged applications.

7. What reports and technical papers have you encountered that discuss these questions?

The research on this effort has considered several of these questions and we have had many discussions with other researchers, but we know of no technical reports or papers that treat these questions in any detail.

The IDES research may be of some interest.

8. What constitutes a failed access attempt? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject? How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool?

The Oracle RDBMS tries to detect attempts to access database objects beyond the subjects discretionary authorization through the "NOT EXISTS" system auditing option and the "WHENEVER NOT SUCCESSFUL" option. The NOT EXISTS system auditing option is designed to detect a user "fishing" for data. If the user names a table that either does not exist or to which he does not have access, this audit option will record the attempt in the audit trail. The WHENEVER NOT SUCCESSFUL option selectively audits unsuccessful attempts to access objects or make use of system privileges.

In MLS Oracle, attempts to bypass MAC should be detected by OS auditing. The Oracle RDBMS is subject to the same MAC as the user and has no privilege to bypass MAC.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

- Use of locks?
- Delete down operations?
- Inference and aggregation attacks?
- Probing by means of integrity constraints?

The TCB subsetting approach used by Oracle is structured so that no covert channels are introduced. In MLS ORACLE the trusted OS will be responsible for enforcing MAC, and therefore any search for exploitation of covert channels should make use of OS auditing tools.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

Not currently.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

The recommended default depends on the needs of each installation or application. By default, no auditing is enabled. A recommended minimum

for a production environment is to audit both successful and unsuccessful logons and "not exists" errors.

12. How does the audit information which can be captured from an interactive user, differ from that which can be captured from an applications program? How does the use of pre-compiled queries affect the audit trail?

There is no difference between audit information for interactive users and applications programs. When stored procedures are available, they will not affect auditability other than to introduce new auditable actions on a new object.

13. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both implicit and explicit, auditable?

MAC and DAC constraints can both be applied to metadata. Metadata access is fully auditable. Data dictionary tables can be audited implicitly by auditing SQL commands that access them. Data dictionary tables are owned by a special database user, and the database does not allow direct grants on dictionary tables. In order for a database user to access a dictionary table, a view must be created, which can be granted and audited.

System Summary: Oracle

The following "interesting notes" and responses to the audit questionnaire are based on documentation, Oracle responses to the questionnaire, a meeting with Oracle, and knowledge of a prototype produced by the NCSC's Secure Relational DBMS project.

Interesting Notes about the Oracle DBMS

Auditing in the Oracle RDBMS takes place at "parse time". This means for instance, that audit records refer to the name of the view referenced as opposed to the query plan that results from "query modification."

Oracle may support data content auditing in a future release.

When a view is created on a table and auditing is set on the table, the view will automatically inherit the auditing option settings of the table. If there are more than one table involved, the auditing on the view is the union of all auditing on the tables. If the table audit is later turned off, the view's auditing remains on, unless specifically turned off.

If a view and a table are created and later audit is turned on the table, then only direct accesses to the table are audited. If auditing is turned on the view and not the table then only accesses through the view will be audited.

B. Notes on Oracle's DAC Policy

Subjects are processes operating on behalf of users.

Objects are tables, views, and sequences.

Authorized users may grant privileges to other individual users directly or to groups of users via the role mechanism.

Database administration are given DBA privilege in version 5.1 and 6.0. In version 7.0, the DBA privilege is provided as a role to which all privileges are granted. The DBA role can be dropped or redefined. Installations can split up administration responsibility into separate and distinct roles, such as security officer, account administrator, etc.

Granting and Revoking

Owners of objects may grant privileges to other users.

Roles are objects that contain privileges. The user creating a role is implicitly granted that role WITH ADMIN OPTIONS. Anyone who has been granted a role may grant/revoke privileges to/form it, even if they were not granted the role WITH ADMIN OPTION. They may also propagate the WITH ADMIN option. Revoking a role has no cascading side effects.

Privileges granted directly can be granted with the WITH GRANT option. Users with the WITH GRANT option or the owner of the object may only revoke the privilege from users if they were the one who originally granted the privilege to that user. If a privilege is revoked, the revoke will cascade.

Security on Views

Views can be value-dependent, time-dependent, context-dependent, and/or user-dependent.

OS Dependencies

Oracle depends on the OS for support for the CONNECT INTERNAL access facilities, protection of storage objects, and for protection of address spaces.

There are two types of "connect internal" access. One to enable a user to perform routine operational functions (e.g., start-up and shutdown), and one for non-routine emergency use (e.g., recovery from a failure that includes the failure of the DAC mechanism). The latter type gives the user very high privilege. The use of the two types is controlled by the OS DAC. Each type is controlled by a separate OS privilege. Oracle checks to make sure that the user's process has been allocated the appropriate privilege before access of the corresponding type is allowed.

C. Notes on the Oracle Architecture for MAC

Oracle is taking the TCB hierarchical subsetting approach.

A database, a collection of permanent storage objects, all passive. A database is a collection of permanent storage objects, all passive. The database includes one or more database files, one or more control files, and two or more redo logs. An instance is a set of active subjects and temporary buffer space. An instance includes an SGA and from zero to five detached processes. An instances is first created and then mounted to a particular database. A database can only be accessed through an instance, and more than one instance can mount the same database simultaneously.

In the MAC architecture, an instance can be mounted on a database if and only if it dominates the security level of the database. If the security level of the instance is not equal to the security level of the database, than the instance is mounted on the database for read-only access. If both are at the same level, then read-write access is possible.

A separate database instance exists for each distinct security level supported by the m-TCB. The m-TCB uniformly labels all aspects of a database instance, including process, memory, and files.

Oracle supports read/write access and read-only access to database instances, but does not support write-only access.

Starting a multi-user database creates a SGA, opens a series of files, and starts five detached processes to coordinate the multi-user operation of the RDBMS. A given instance will be started at the first R/W reference to that instance.

To support the read-only access a non-shared global area will be created in the process address space for any database instance to which a subject requests read-only access.

Relational Technology Inc.

Product: Multi-Level Secure Ingres

Date of Meeting: August 16, 1989

Point of Contact: Robert McCord
Jackie McAlexander

Questionnaire Author: N/A

System Summary Author: Trusted Information System, Inc.

Reference:

"Multi-Level Secure Ingres Informal Security Policy Model," Relational Technology, Inc., Alameda, California, draft July 31, 1989.

System Summary: Ingres

I. Status

Product is underdevelopment.

II. Architecture

Ingres is a trusted database that runs on a trusted operating system (B1 or higher). This trusted Ingres does not directly support any network or cluster DBMS functions.

III. Security Policy

The Ingres subjects are identical to the underlying operating system subjects (i.e., the process/domain pair acting on behalf of a specific Ingres user).

The objects are database rows, tables, views, and SQL procedures. All other objects that are not within an Ingres database are controlled by the underlying trusted operating system.

Discretionary access controls is based on an ownership model for databases and tables. When a user creates a database, he/she becomes the owner. Within a database, there are discretionary controls on named objects: tables, views, and SQL procedures. The creator of object is the owner. For tables, there are additional associated objects, such as indexes, that are implicitly owned by the creator of the table. The owner of an object has privilege to use the SQL Grant statement to authorized access to that object. Ingres supports six access modes for tables and views: SELECT, INSERT, UPDATE, DELETE, COPY_INTO, and COPY_FROM.

Mandatory access controls are enforced by using the row as the single labeled storage object. Each row of a table is independently labeled when it is stored in a database. The security label is defined as a normal column within a table. The access modes for a row are SELECT, UPDATE, INSERT, and DELETE. All of the resources managed by Ingres are stored as rows in some table.

The mandatory access controls do not automatically polyinstantiate rows. However, intentional INSERT polyinstantiation can be imposed by a database administrator when tables are defined. If security label is included as a key column of all UNIQUE indexes, then the rows will be effectively polyinstantiated when they are inserted by users with different security labels.

IV. Auditing

Audit records are generated for the operating system and Ingres security events independently. Ingres chose to store the audit log in a file (different than the operating system audit file) and not a table in the database. This decision was primarily to increase performance, and also because the audit log requires auditing frequently and they have existing tools to archive files.

Ingres logs the success or failure of the following events:

Database (create, open, close, destroy)
Application (create, delete, execute)
Procedure (create, delete, execute)
Table (open, create, delete, modify, index)
Location (create, delete, extend)
View (create, open, close, delete)
Record (select, insert, delete, update, copyinto, copyfrom, scan, escalte)

Ingres provides an auditing tool, auditsec, that gives the user reviewing the audit log a limited set of query-like search capabilities.

V. Other Topics of Interest

Two Versions

RTI plans to have a B1 level version in which DAC/MAC are in a single protection domain. A B2 version is planned in which MAC is in a separate protection domain.

Two Approaches Used

1. TCB subset approach (i.e., DBMS is a MAC kernel)

2. Balanced assurance approach (i.e., DAC = C2, MAC = B division).

SRI

Product: SEAVIEW prototype

Date of Meeting: July, 1989

Point of Contract: Teresa Lunt

Questionnaire Author: N/A

System Summary Author: N/A

NOTE: Since the researchers at SRI had not fully considered the audit issues specifically in the SEAVIEW prototype, the discussions held at SRI focused primarily on general research and intrusion detection techniques. It was this meeting that the need for auditing at multiple levels of abstraction was first brought out. There is no questionnaire or system write-up associated with this interview.

SYBASE/TRW

Product: SYBASE Secure SQL Server

Date of Meeting(s): SYBASE, July 10-11, 1989
TRW, June 19, 1989

Point of Contact: Helena Winkler-Parenty (SYBASE)
Tom Winkler-Parenty (SYBASE)
Ed Sturms (TRW)

Questionnaire Author: Trusted Information System, Inc.

System Summary Author: Trusted Information System, Inc.

References:

1. Rougeau, P.A., and E.D. Sturms, "The SYBASE Secure Dataserver: A Solution to the Multilevel Secure DBMS Problem," Proceeding of the 10th National Computer Security Conference, Baltimore, Maryland, September 1987.

2. O'Brien M.F., and E.D. Sturms, "MLS Implementation Using the SYBASE Secure SQL Server," TRW Federal Systems Group.

3. The SYBASE Secure SQL Server, Release 1.0 Beta, 1989.

4. SYBASE Secure SQL Server, Version 1.0, 1988.

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current practice and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

The SYBASE system supports three security-relevant roles: User, System Administrator (SA), and System Security Officer (SSO). SA and SSO are allowed to invoke special commands.

The following events are always audited.

- All logon/logoff attempts. (Users must log onto the DBMS server after logging onto the host OS)
- All server boot-up events
- All commands performed at the "trusted I/F" (special terminals and software). This includes all special SA commands and User T.I. commands like grant and revoke, which have been removed from SQL.
- Any (system) "integrity violation" such as the arrival of SQL grant and revoke commands into the execution domain, any query plan errors, domain enforcement errors, any detected corruption of data, etc.
- All attempts to violate DAC restrictions on secondary objects (tables and databases).
- Any attempted operation, other than a select or a reference in a where clause, that affects the security label. Only the SSO is allowed to alter labels, and this is only allowed via a write-down procedure.

All accesses to individual databases, or individual tables may be audited selectively. Auditing options (defaults) for accesses to future objects may be established in advance. For each database, audit can be turned on or off according to the following criteria.

- For each table, either non-select operations, select operations, or both may be audited. This appears to mean that one audit record is stored for each reference to a specified table, regardless of the number of row references implied.

- To obtain greater detail, each reference to any row in a specified table can be audited; auditing can be enabled for either non-select or select operations or both. In this case, the row ID (RID) of each referenced row is stored in the audit log. (Under some circumstances, RIDs may be replaced; therefore a RID does not uniquely identify a row.) In addition, it is possible to specify a MAC level threshold at or above which all row accesses are to be audited.

Audit can be turned on or off on a per-user basis, as well.

- Access to individual tables by specified users can be audited, or row-level access by specified users can be audited.

- Each keystroke (actually, each SQL packet arriving at the DBMS) from specified users can be captured in the audit log. Presumably, the keystrokes contain the text of all queries.

2. For each of the above, what information can be captured about the event?

The audit trail is a relation consisting of the following attributes (some may have blank values):

- security label (every relation has this attribute) always set to system high. Note that the user's session level is stored as a different attribute (below).
- query number (Several entries may be generated for a single query; this field is used to tie them together.)
- time stamp
- login name
- session level
- transaction ID
- type of operation (select, insert, delete, create,...)
- success/failure
- SSO console message number (sequence number)
- database identifier
- object identifier (table, view, stored procedure)
- security label of table
- RID (or blank)
- security label of row
- keystrokes (optional)

3. What special privileges are required for access to the audit control mechanism and the audit log? Is the log single-level or multi-level?

Only the SSO can access the audit control switches and the audit log. The audit log is stored system high.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the O.S. audit log? Are each of these separate? Can they be correlated easily?

Each of these is separate, but they have been designed specifically to be correlatable. RIDs and transaction IDs may be found in both the rollback log and the audit trail. Normally only the database owner (DBO) has the privilege needed to rollback a database, though he can reassign this privilege to another person. The DBO can dump the rollback log to a tape, but the tape must be mounted and dismounted by the operator. It is possible for the tape to be classified higher than the authorization level of the DBO. The SA and SSO are assumed to be cleared to system high. Because of separation of duty, it may be necessary that the DBO, SA, and SSO work together to "reconstruct a crime" using the rollback log and the audit trail.

Keystroke information might be of greater value than capturing and analyzing RIDs of retrieved entries and attempting to correlate them with the rollback log.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

The audit trail is an ordinary database relation. The intended analysis tools are the query language and other database facilities provided for the general user population, excluding those for modification. However, since some of these facilities in the B2 system under development (e.g., the query language parser/compiler) are untrusted, the auditor cannot fully trust the results of a query on the audit trail. SYBASE also provides a hardcopy audit log, so that (in theory) an auditor could cross-check the results of an audit query against a trustworthy representation of the audit trail.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

Anyone can easily write an application that runs off sysaudit. Transact SQL or SQL toolset can be used.

7. What reports and technical papers have you encountered that discuss these questions?

None.

8. What constitutes a failed access attempt? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject? How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool?

There are three cases. Attempts to reference tables or databases in ways that violate DAC constraints are always audited. Attempts to violate MAC on databases and tables are audited as failed attempts, although the user receives an indicator that the object doesn't exist. Attempts to retrieve rows that are beyond the MAC

authorization of the session are NOT audited as a failed attempts, and simply do not appear in the user's response.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

- Use of locks?
- Delete down operations?
- Inference and aggregation attacks?
- Probing by means of integrity constraints?

This is under study as part of the covert channel analysis for the B2 target system. TRW and SYBASE may have more information in a month or so. Incidently, the granularity of locks for mutual exclusion is that there is one lock per 2KB page.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

No.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

At present there are no recommended defaults beyond those events that are always audited (not selectable).

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

Use of precompiled queries "stored procedures" and views DOES affect the keystroke audit data, and could potentially be used to disguise actions from an auditor. This would require the auditor to examine the table and row level audit data in addition to the keystroke data.

Neither use of views, nor use of pre-compiled queries, has any effect on the other forms of information captured in the audit trail.

13. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both explicit and implicit, auditable?

It appears that both MAC and DAC constraints can be applied to metadata. Direct accesses to metadata are auditable, but indirect accesses are not.

Other Notes and Questions

Operations attempted during transactions are auditable regardless of whether the transaction is committed or not. However, the beginning and failure or successful completion of a transaction will not appear as events in the audit log, except that keystroke audit data may reveal use of explicit SQL statements to start or stop transactions.

Changes in DAC constraints take place on the next query execution. That is, there is minimal delay in binding.

The rationale for features chosen was based on the development team's own intuition and experience and was not driven by specific guidance from customers, or government agencies.

The SYBASE B2 version of TCB consists of two domains - execution and I/O. All B2 auditing is handled by the execution domain becaue it deals with higher level semantics and can capture more meaningful audit information. Within the execution domain, audit info is captured at a number of different stages of processing. Audit data is forwarded to an audit logger process via a mailbox (buffer).

System Summary: SYBASE Secure SQL Server

I. Status

The targeted B1 version of the SYBASE Secure SQL Server database management software will be shipped in December, 1989. SYBASE is also working on the development of a B2 version of the Secure SQL Server.

II. Architecture

The B1 version is secure running either on an untrusted OS or a trusted OS, and can be ported to a secure UNIX OS or another trusted OS, such as SEVMS. The SQL Server enforces its own security policy, providing mandatory and discretionary access controls, identification and authentication, and auditing. The operating system is used primarily for disk management. The B2 version eliminates the need for a secure operating system and will run on bare hardware.

III. Security Policy

The objects known to the TCB are rows, tables and databases. Subjects are users and processes.

Rows are subject to mandatory access control. Each row is an independent security object and has a security label stored as part of the row. Tables and databases are labeled at creation time with the login security level of the user. The definition for each table and database is stored in system tables. Each row defining the table or database is labeled at the security of the object. This provides the effect that users will be unaware of the existence of any tables or databases they are not authorized to see. The label of the row in the sysobjects system table serves as a minimum access level for the table or database. Therefore, users granted discretionary access to a table or database must be logged in at or above the level at which that object was created in order to know that the object and its attributes exist.

Databases and tables are subject to both discretionary and mandatory access controls. For each database, users can be granted permission to use the database or to create tables in it. To access a database or table, the user's login security level must dominate the level of the database or table.

The security policy allows a user to modify rows that are labeled at the user's login security level. If the row is labeled below the user's login security level, the existing row is changed and a new, modified row is inserted that has the user's login security level as its label (polyinstantiation) Additionally, a user is not allowed to delete a row unless the user's security level exactly matches the label of the data. SYBASE does provide a relaxation property which when turned on, turns polyinstantiation off and allows user's to "delete down".

All inserted rows are labeled at the user's login security level.

A user may retrieve rows at or below his or her login security level. Changes in DAC constraints take place on the next query execution. That is, there is minimal delay in binding.

To provide additional integrity protection against hardware and software failures, an algebraic computation (cyclic redundancy check) is performed on each page and stored in the page header.

IV. Auditing

See survey.

V. Other Topics of Interest

1. Trusted Interface

There are three Trusted Interfaces in the Secure SQL Server, all connected to the TCB by trusted paths. They are:

- System Security Officer Trusted Interface, used only by SSO's and where all SSO-specific functions are performed.
- User Trusted Interface, where System Administrators perform many of their functions, object owners manage discretionary access permissions (grant and revoke), and all users change their passwords.
- Tape Console Trusted Interface, which provides a simple interface for dumping to and loading from tapes.

In addition, the Audit Trail is a trusted path. It is usually a printing terminal, to which all audit messages are sent.

2. Data Flow

In the B2 Secure SQL Server, each SQL query is passed over the network to the untrusted DBMS code where it is parsed and compiled into a query plan. The query plan is passed from the untrusted code into the TCB where a copy of it is made so that it cannot be further altered by the untrusted code. The TCB then performs procedure validation and access control mediation (MAC and DAC) on the database being used and on any tables accessed. When data is written to a page, that page's CRC is updated. When data is retrieved from a page, the page CRC is checked and MAC checks are performed on any row retrieved. The query is executed, and appropriately labeled rows are returned to the host. Row security labels returned to the host are advisory.

TANDEM

Product: N/A

Date of Meeting: July 13, 1989

Point of Contact: Claudia Ruoti

Questionnaire Author: Trusted Information System, Inc.

System Summary: N/A

Reference: None

Security Officer Questionnaire

The objective of this questionnaire is to gather information about current usefulness of audit trails.

1. What information does the Security Officer receive in an audit trail?

   Several different audit trails are available for review by the Security Administration (SA) or EDP Auditor. These include operator activity, system access, security administration activity logs and application recovery logs (TMF).

   What type of system (i.e., operating system, database...)?

   The type of systems running was not elaborated upon, but it appears that they are using both database audit trails and operating system audit trails depending upon the application being used.

2. How is this information used?

   Audit trails are examined to determine that only authorized programs are running, that only authorized users are accessing programs in an appropriate manner to try to find unauthorized accesses and users; and for verifying compliance with policy and procedures.

3. *Is the information* gathered adequate to meet the required needs?

   No. The audit records were too operationally oriented. While the data is available, it is in many different formats and is therefore not user friendly as it requires a significant amount of technical support or expertise to consolidate or extract information.

   If not, what information would be adequate?

   The capture of screen activity images in addition to before and after records in the TMF log could prove helpful as an investigative tool.

   She would also like to be able to better protect the audit trail from disclosure.

4. Are tools used in reviewing the audit trail?

   Tools are available for reviewing audit trails but they are cumbersome, require technical knowledge of the "process" which generated it, and are of limited benefit.

   If so, how useful are the tools?

   They are somewhat useful.

What kind of tool would be useful?

A tool that maintains the integrity and security of audit trail information when downloading it to some other medium. She is concerned with, at a minimum, maintaining the integrity of the audit trail because the dump can be fouled and the auditor must be able to trust the audit trail. A tool report writing/analysis mechanism that non-technical users can utilize effectively.

5. What is the purpose for auditing?

They audit to ensure the integrity of the operating environment is maintained. EDP Auditors examine the general environment (technical, operational and administrative) as well as the application environment. General controls are established to maintain the general environment. The general controls must be reliable so that the application programs can be trusted. An application may be very well controlled but if the general controls are corrupt, then none of the information from the application program or recovery actions performed by the application program can be trusted. General controls require characteristics such as good policies and procedures, and segregation of duties. Tandem examines the general controls on a regular basis looking at the following items:

1) compliance with policy and procedures,
2) physical environment,
3) the integrity of the code,
4) the segregation of duties,
5) determining what programs are running when *(correct time)*,
6) ensuring only authorized programs running,
7) ensuring only authorized users accessing programs in an appropriate manner, and
8) examining automated tools.

Applications logs(TMF) are used for recovery. They will need before and after images. Recoverability is critical to an auditor. Application reviews must occur periodically, especially when the application is programmable, to ascertain that the security policy has not been compromised. The application could even determine how long specific tapes should be kept. An application that lets more than two people on the same terminal or allows someone to perform an action such as raising one's own security clearance is suspect.

Is auditing useful or necessary? Yes.

6. When is the audit trail reviewed?

System level and security log reviews should be ongoing daily at a minimum.

Application logs should be reviewed daily for integrity. The frequency of contents review and is dependent on the security and sensitivity of the application itself.

Teradata

Product: BBC/1012

Date of Meeting: N/A

Point of Contract: Jim Pierce

Questionnaire Author: Teradata

System Summary Author: N/A

Reference: None

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current practice and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

   All user-initiated events are auditable. That is, logon, logoff, and each statement submitted by a user. Logon and logoff are always audited while other events are selectively audited. The selection criteria can be used name, object name, label of objects and statement type executed or any combination of these four. Audit entries can be selectively generated either on denials only or on all events meeting the specified criteria.

2. For each of the above, what information can be captured about the event?

   The same basic data is captured for all events. It includes:

   Event date and time
   Logon date and time
   User internal and external identifiers - name and account
   Internal identifier for session under which user is logged on
   Source of logon for the session
   Label of the session
   Result of DAC or MAC execution, i.e., denial or success
   Frequency or count of events during session
   Object access - includes database, table, and column names
   Name of object's owner
   Label of object accessed
   Statement type executed

   Optionally the complete text of the statement submitted (DAC only). Obviously, if it is a logon of logoff there is no object information.

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log? Is the log single level or multi-level?

   Specification of audit criteria is reserved to the Super User in the system. The Super User can grant privilege for audit specification to other users. Granting that privilege is an auditable event. Access to the audit log is reserved to the Super User in the system. The Super User can grant access to the log to other users. Granting such access is an auditable event. Each audit entry contains the label of the subject and object for which the event was audited. However, from a system standpoint, the labels in the audit entries are data not labels. From a system standpoint and MAC, the entries are single level.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

   There is no relationship between the audit log and recovery journals other than both are created from the same event. There is no relationship between the DBMS audit log and the OS audit log. The DBMS audit log and recovery journal are separate. The DBMS and OS logs are separate. If the OS log contains date, time, and user identity then it should be possible to coordinate entries on the two logs. The DBMS recovery journal contains session identifiers so it should be possible to coordinate entries. However, since audit entries are optional by user and access type for an object while recovery entries are not, coordinating them may not make sense. Additionally, a recovery journal is not directly user accessible. They can use it for recovery, but cannot access individual entries as they are internally formatted for faster recovery.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

   Access to the audit log is done through the normal DBMS functions. There are no special tools provided. If the functions of the DBMS are rich enough for application use they should suffice for audit entry access.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

   The language for audit criteria could be extended to define some action for the DBMS to take when some identifiable event occurs. The DBMS now simply generates an audit entry. The action could be to log off the offending user, send a message to a mailbox, etc. The circumstances under which this action was to be taken would have to be fairly simple. The DBMS doesn't have any AI capabilities.

7. What constitutes a failed access attempt? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject? How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool?

   A failed access attempt is a violation of the system implemented DAC or MAC policy. The DBMS does include authorizations for access by subject to an object for specific actions. Each audit entry contains a field indication whether the entry was generated for a failed or successful event.

8. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

   Being a novice at security, I don't understand these questions. Rather than give misleading answers, I will abstain completely.

9. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

   The DBMS does not contain such a facility. See 6 above.

10. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

    Since the level of auditing really depends on the data controlled by the DBMS and the application of that data, we do not suggest any minimum auditing of object accesses. We always generate audit entries for logons and logoffs. If I were to recommend further minimum levels, I would obviously start with auditing those events that control the security aspects of the system; e.g., granting and revoking access privileges, creating new users, changing user passwords, etc.

11. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

    There is no difference in auditing of interactive or batch users. Auditing is by user name, object, etc., not how the user interacts with the DBMS. The use of precomplied queries does not change audit functions. It does complicate their implementation. We apply the security policy to all queries no matter whether they are precomplied or not. If a complied query is 'saved' we maintain structures with that compilation that are then modified based on the user executing the saved compilation to cause application of the policy based on the circumstances at the time of execution not at the time of compilation.

12. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both implicit and explicit, auditable?

    MAC and DAC constraints can be associated with metadata. Audit generation is selectable by object, user, and action. Metadata entries are identifiable objects, therefore access to the entries and auditing of those accesses can be specified. Since write access to these entries is a byproduct of some other user action, the list of events that can be specified for metadata objects is different than the events for user created objects. Read access to metadata entries can be controlled like any user created objects. Metadata entries are deemed to be 'owned' by the Super User in the system. That user controls access to metadata entries.

13. What reports and technical papers have you encountered that discuss these questions?

    None, although I haven't really looked.

TRW

Product: ASD_Views

Date of Meeting: N/A

Point of Contract: Tom Hinke

Questionnaire Author: TRW

System Summary Author: N/A

Reference: N/A

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current research and practice, and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable?

Open table, user login, and all sso functions.

Which of these are selectively auditable?

Successful open table function. The failed table opens, user login and all sso functions are always being audited.

2. For each of the above, what information can be captured about the event?

Open table: user name, user security level, success or failure, table name, reading or writing to table, table owner name.

Sso functions:

reclassify: sso name, sso security level, success or failure, reclassified table name, new security level for the table's tuples, tuple qualification clause.

aql query: sso name, sso security level, success or failure, the aql statement.

add user: sso name, sso security level, success or failure, user name added.

delete user: sso name, sso security level, success or failure, user name deleted.

list users: sso name, sso security level, success or failure.

list all audited results by user: sso name, sso security level, success or failure, user name.

list all audited results by event: sso name, sso security level, success or failure, event type.

list all audited results by table: sso name, sso security level, success or failure, table name.

list all audited results: sso name, sso security level, success or failure.

set audit option (turn on or off): sso name, sso security level, success or failure, on *or off.*
When turned off, only successful table opens are turned off, all others are still on.

sso error (sso tried to do something not supported by sso server): sso name, sso security level, what sso tried to do.

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log?

Sso is the only one who can turn on/off audit control.

He can do so at any security level. The audit log is created by the DBMS kernel, upon DBMS start-up, at system high. Only the system high sso can read the audit log.

Is the log single level or multi-level?

Single. System high.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal?

Don't know. ASD's recovery feature was not implemented yet. This issue has never been discussed.

Between the DBMS audit log and the OS audit log?

DBMS' audit log is a system high file on the os. Don't know about the os audit log. ASOS doesn't know about the DBMS audit log yet. The DBMS never thought about the os audit log, and never worries about the os audit log when it's auditing. So, probably no relationship for now.

Are each of these separate?

Yes.

Can they be correlated easily?

Don't know. Never discussed this with ASOS.

5. What automated tools are provided for analysis of the audit log?

A graphic sso interface for sso to list the audited results with options. E.g., by user, by table, by event, etc.

What additional tools would be useful?

Turn on/off audit events by event type.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

Don't know. Never thought about this.

7. What reports and technical papers have you encountered that discuss these questions? (This should have been the last question, but was left as question 7 for consistency with earlier versions of the questionnaire.)

Henry - none. Tom -? Cristi -? Amy -?

8. What constitutes a failed access attempt?

Open table: invalid table name, discretionary access to table denied.

User login: anonymous user.

Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject?

Of course, Bell-Lapadula is in effect here - can't read up, can't write down. Plus, discretionary control on each table for each authorized table user.

How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool?

Audit log entries for these events would be with status FAILED. So the tool can scan the audit log file for this status.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)?

????

How might these be represented in the audit log?

Need to answer the above question before answering this one.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

No, not right now.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

As of now, the DBMS only provided option to select audit of successful table opens on or off, it doesn't allow a subset of events not to be audited (or audited). But, the minimum set of info is what the DBMS has now - user login, open table, sso functions.

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program?

No difference, an applications program still needs to log into the DBMS as an user, and do all DBMS queries on behalf to an user.

How does the use of pre-compiled queries affect the audit trail?

???

13. Can MAC and DAC constraints be associated with metadata?

Yes for MAC. Yes for DAC and not sure of affect a user with no access.

Are all metadata accesses, both implicit and explicit, auditable?

???

14. At what stages in the processing of a query (parsing, compilation, optimization, execution, etc.) can audit capture occur?

It can occur any where, but in our case, it's in execution, which is in the DBMS kernel.

To what extent is the choice of capture points constrained by the DBMS architecture?

For our case, capture points can be any where as long as they are in the DBMS kernel, because the audit log is a system high file on the os, and the DBMS kernel is the only trusted process in the DBMS architecture.

How does the choice of capture points affect the credibility and usefulness of the audit trail?

???

15. What audit information is captured for transactions that are never committed?

Not captured in our case.

Is it possible for a user to evade auditing by aborting transactions that include suspicious access attempts?

No, not in our case.

16. With respect to on-going sessions, when do changes in DAC constraints and view definitions take effect?

????

If a significant delay is possible, how is the time-of-effect reflected or inferable from the audit log?

???

17. Other interesting issues, ideas, suggestions?

????

UNYSIS

Product: Secure Distributed Database Management System (SD-DBMS)

Date of Meeting: June 19, 1989

Point of Contact: Cathy McCollum (UNYSIS)
LouAnna Notargiacoma (formally of UNYSIS)

Questionnaire Author: Trusted Information System, Inc.

System Summary Author: Trusted Information System, Inc.

Reference:

1. *Secure Distributed Database Management System (SDDBMS)*, Final Report, Volume 1: Architecture Security Policy/Formal Model DTLS, UNISYS for Rome Air Development Center, Contract No. F30602-87-C-0154, February 15, 1989.

DBMS Audit Questionnaire

The objective of this questionnaire is to gather information about current practice and to collect suggestions to support the development of MLS DBMS auditing requirements.

1. What DBMS events are auditable? Which of these are selectively auditable?

The SD-DBM system provides for three different auditor roles: the DBA or database administrator/owner, the SSO or System Security Officer, and the Auditor. Each role is responsible for auditing one of three disjoint sets of auditable actions. For instance the DBA audits user actions performed upon his/her respective database, while the SSO audits the actions of the DBAs, such as creating a database. Finally the Auditor audits the actions of the SSO. Specifics on the events audited by each role can be found in the DTLS; copies of these pages are attached.

A condition clause in the audit statement indicates when an action should be audited (success/failure/always).

All audit events are selectively audited.

2. For each of the above, what information can be captured about the event?

The information collected for a given event was not yet specified. It had not been considered whether it made sense to capture the text of queries.

3. What special privileges or clearances are required for access to the audit control mechanism and the audit log? Is the log single level or multi-level?

Three audit files are created and stored at different levels. The audit trail created by the DBA is stored at the high water mark of the database. Note that the user performing issuing this audit statement must be an authorized DBA and the owner of the database. The audit trails created by the SSO and Auditor are both stored at syshigh.

4. What is the relationship between the DBMS audit log and the DBMS rollback-recovery journal? Between the DBMS audit log and the OS audit log? Are each of these separate? Can they be correlated easily?

These issues have not been investigated.

5. What automated tools are provided for analysis of the audit log? What additional tools would be useful?

None at this time.

6. In what way could automated intrusion detection techniques be usefully integrated with the auditing mechanism or off-line audit analysis tools?

N/A

7. What reports and technical papers have you encountered that discuss these questions

None.

8. What constitutes a failed access attempt? Could (does) the TDBMS detect attempts to access data beyond the authorization of the subject? How might such attempts be represented in the audit log such that they could be detected by an off-line analysis tool?

A failed query is a query that does not pass the validation portion of the system because the user is unable to see the resultant view. It is not even passed to the respective database. A suggestion was made that it might be useful to set bounds indicating the number of failed accesses that equal a potential security violation. Once this threshold is reached an audit record would be generated.

9. What constitutes an event "that may be used in the exploitation of covert storage channels" (TCSEC)? How might these be represented in the audit log?

- Use of locks
- Delete down operations
- Inference and aggregation attacks
- Probing by means of integrity constraints

This had not been explored from an auditing standpoint.

10. Are there any capabilities for automatically alerting the security officer that an attack might be underway?

None at this time.

11. Given that audit capture is typically selectable, what is a recommended default (or minimum) set of information that should be captured?

This had not been considered yet.

12. How does the audit information capturable from an interactive user differ from that capturable from an applications program? How does the use of pre-compiled queries affect the audit trail?

Ad hoc queries from an interactive user look the same as an application program.

13. Can MAC and DAC constraints be associated with metadata? Are all metadata accesses, both implicit and explicit, auditable?

All metadata is separated into single level files. The OS mediates access to these files and may audit these accesses.

System Summary: Secure Distributed DBMS

I. Status

This effort was a research project to build a prototype of a Distributed DBMS. The prototype was built and there has been no further work performed.

II. Architecture

Unisys's secure distributed DBMS (SD-DBMS) architecture consists of three types of components: user programs, data manager, and back-end DBMS (one untrusted DBMS per security level supported).

a. A User program is a user interface or application program permitted to issue queries against a multi-level database. User programs can be trusted (i.e., multilevel) or untrusted (i.e., single level). Trusted user programs are permitted to issue queries at multiple security levels and receive multi-level results.

b. The Data Manager is a trusted component that performs reference monitor functions for SD-DBMS.

c. The Back-end DBMS are untrusted single-level relational DBMS used to store and process portions of the multi-level database. The SD-DBMS stores multi-level relations by horizontally partitioning them into single-level fragments, which are then stored under appropriate back-end DBMS.

III. Security Policy

The subjects are user programs. The objects are tuples in multilevel relations, user queries, access views, and schemes.

The SD-DBMS uses access views as a mechanism to enforce discretionary access control. An access view is a virtual relation derived from base relations. Subjects are not permitted to directly access base relations. All access to base relations must go through access views.

Mandatory security in SD-DBMS is the association of access classes with the individual tuples in base relations (i.e., the lowest level objects in the system). Subjects can access these tuples through operations on access views defined on base relations.

IV. Auditing

The SD-DBMS provides for three different auditor roles: the DBA or data administrator/owner, the SSO or System Security Officer, and the Auditor. Each role is responsible for auditing one of three disjoint sets of auditable actions. For instance

the DBA audits actions performed upon his/her respective database, while the SSO audits the actions of the DBAs, such as creating a database. Finally the Auditor audits the actions of the SSO.

V. Other Topics of Interest

When a user creates a multi-level relation, the system creates a set of single-level fragments in which to store the relations. To retrieve data from a multi-level database, subjects (User Programs) submit queries to the Data Manager. The Data Manager decomposes each query into a sequence of subqueries that operate on single-level fragments. Once a query is decomposed into subqueries, each of the subqueries is executed on the back-end DBMS having the same security class as its result. Since queries executed on the high DBMS often require access to low data, the SD-DBMS supports the transmission of data from the low to the high DBMS. To assure that no data flows in the opposite direction, all such transfers are constrained to go through the reference monitor.

XEROX

Product: N/A

Date of Meeting: August 8, 1989

Points of Contact: David Goldhirsch
Upen "Sharma" Chakravarthy

Meeting Notes Author: Trusted Information System, Inc.

Questionnaire Author: N/A

System Summary Author: N/A

Reference: N/A

NOTE: Since the researchers at Xerox had not specifically investigated auditing in database systems, the meeting described by this write-up took the form of an informal discussion rather than an interview structured to follow the survey questionnaire. As a result, this write-up consists of informal meeting notes, and does not follow the format used for the other write-ups.

HiPAC: A research effort to investigate use of event triggers for integrity violations, data arrival (sensor updates), creation of derived data (corporate snapshots) and view consistency. Used an extensible research DBMS called "Probe" as basis for experiments.

Could probably use trigger mechanism for audit data capture, however, investigating auditing was not part of Xerox's reseach.

Issue: If audit trail (relation) is huge, queries on it will take forever unless its storage is optimized for certain types of retrieval. Optimization for retrieve involves definition and maintenance of secondary/tertiary indices; maintenance of these is likely to degrade performance during audit capture.

Issue: What happens to audit records of a complex transaction fails (is backed out)? Are the audit records kept and followed by an indication that the transaction failed? (probably) Or are the audit records purged?

Rules/Triggers:

An Example: At the end of each buy or sell transaction, the stock portfolio manager must have a positive account balance. (During the transaction, the balance may temporarily be less than $0.00.) If this rule is violated, some specified action (possibly another transaction) will be triggered.

Financial and accounting organizations are very concerned with thresholds, and might only want to collect audit data when certain thresholds are reached; for example, audit all checks written for amounts over $100K. (Note: It may be useful to call this "value-based auditing". This type of capture appears to be potentially more selective than that suggested by other research or development groups.)

Rules may be transactions that "range over" (are triggered by) not only changes in the application's data, but changes or values in the "transaction manager's" control info including the lock tables, transaction IDs, process IDs, etc. Marv Schaefer's example of falsifying a credit rating database by modifying an alias table provides another example: the alias table can be considered as control info that might need to be referenced in a set of audit rules.

Dave Goldhirsch has researched use of "cascadable" rule mechanism and concluded that its use has the potential to cause difficult problems. (The action triggered by one rule causes other rules to be triggered.) He believes that cascading rules must be defined "gingerly;" great caution is required to avoid undesirable or unintentional rule interactions.

Potentially, rule-based audit capture could be more selective, reducing chances of being overrun by useless audit data. The potential cost is the execution overhead to check the rule base for applicable rules at the start and/or end of each transaction. The overhead depends on the complexity and interdependence of rules. Sharma proposed using a rule-based mechanism to figure out exactly what audit data is of

interest, then replacing it by a different, less-flexible mechanism, such as a fixed set of on/off switches, to increase the efficiency of capture. Or, it may be useful to have a hybrid approach providing a fixed set of switches for commonly needed data and a rule based mechanism for auditing based on a few more complex patterns of interesting behavior.

If rules can reference concurrency locks, one might be able to use them to capture only the lock usage that fits specific patterns indicative of possible exploitation of covert channels. (This assumes that such patterns can be identified.) Since locks are manipulated extensively in most multi-user and distributed databases, a simple on/off switch to record lock usage would probably generate an overwhelming amount of data! Perhaps this is a good example of how a rule could be used to zero in on only very specific usage patterns.

Transaction log -
Typically snapshots are saved, but not the log of what happened in between; typically the log is only used to keep track of what happened since the last snapshot. Once a new snapshot is established, the log is purged and started over. However, there are several counter-examples in which, for legal or other reasons, the log must be kept over a long period of time (airline reservation system, W. German privacy laws). Snapshots are "consistency firewalls," capturing the state of the DB at an instant when the DB is temporarily in a consistent state.

Sharma recommended looking for a single, minimal representation that includes both the transaction recovery log and the security audit trail. Note that this idea seems to be expressed in one of Sushil JaJodia's recent proposals.

Papers/sources on this subject: Intrusion detection and inference/aggregation work from SRI, Denning, Morgenstern.

Possible covert channels include all shared system-related data and metadata including integrity constraints, system-internal data structures, locks, etc.

Regarding the Denning "tracker problem" (inference/aggregation) - To detect attacks of this nature, a history of QUERIES must be maintained. Implication is that the audit mechanism must capture, or be able to reconstruct, the query text.

Suggestion for audit analysis tools :

Supplement relational query capability with an inferencing mechanism.

- To support analysis of inference attacks, use inference mechanism to figure out what an attacker might be able to infer.
- To support complex analyses like "Who might have seen the false positive aids test result for John Doe?" Note that there are many possible paths via views, counts, sums, interacting transactions,

etc., that may have revealed directly, indirectly, or propagated John's false positive test result.

Another suggestion:

To detect inference/aggregation attacks, define views (derived data) that represent info that can be inferred. Then compare user queries to the derived data view. Overlap indicates data that the user could have inferred from his/her queries. A recent paper from Hinke discusses this general topic.

Bibliography

[AFSB83] Schaefer, M., Air Force Studies Board, "Multilevel Data Management Security," National Research Council, National Academy Press, Washington, D.C., 1983.

[AUDIT87] A Guide to Understanding Audit in Trusted Systems, NCSC-TG-001, National Computer Security Center, July 1987.

[BUCZ89] Buczkowski, L.J. "Database Inference Controller," *Proceedings of the 1989 IFIP Workshop on Database Security*, Monterey, California, September 1989.

[CLARK87] Clark, D.D., and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proceedings of the 1987 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, April 1987.

[CLBR89] Crocker, S., J.R. Landauer, T.C.V. Benzel, and T. Redmond, "Formal Policies for Trusted Processes," *Proceedings of The Computer Security Foundations Workshop II*, Franconia, New Hampshire, June 1989.

[CROCK89] Crocker, S., and M. Pozzo, "A Proposal for a Verification - Based Virus Filter," *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, Oakland California, May 1989.

[DEN82] Denning, D.E., *Cryptography and Data Security*, Addison-Wesley, Reading, Massachusetts, 1982.

[DEN83] Denning, D.E. and J. Schlörer, "Inference Controls for Statistical Databases," *Computer Magazine*, July 1983.

[DEN85a] Denning, D.E. and P.G. Neumann, *Requirements and Model for IDES A Real Time Intrusion Detection Expert System*, Finial Report 6169, SRI International, Menlo Park, California, August 1985.

[DEN85b] Denning, D.E. "Commutative Filters for Reducing Inference Threats in Multilevel Database Systems," *Proceedings of the 1985 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, April 1985.

[DEN87] Denning, D.E., S.G. Akl, M. Heckman, T.F. Lunt, M. Morgenstern, P.G. Neumann, and R.R. Schell. "Views for Multilevel Database Security," *IEEE Transactions on Software Engineering*, 13(2), February 1987.

[DEN88] Denning, D.E., and T.F. Lunt, "The Sea View Security Model," *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, April 1988.

[GROH76] Grohn, M.J., *A Model of a Protected Data Management System*, ESD-TR-76-289, I.P. Sharp Associates, Ltd., June 1976.

[H-S75] Hinke, T.H., and M. Schaefer, "Secure Data Management System," RADC-TR-75-266, November 1975.

[HIN88] Hinke, T.H., "Inference Aggregation Detection In Database Management Systems," *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, April 1988.

[HOSM89] Hosmer, H.H., "Handling Security Violations Within An Integrity Lock MLS DBMS," *Proceeding of the 1989 IFIP Workshop on Database Security*, Monterey, California, September 1989.

[JAJ89] Jajodia, S., S.K. Gadia, G. Bhargava, and E.H. Sibley, "Audit Trail Organization in Relational Databases," *Proceeding of the 1989 IFIP Workshop on Database Security*, Monterey, California, September 1989.

[KNOD88] Knode, R.B., and R.A. Hunt, "Making Databases Secure with TRUDATA Technology," *Proceedings of the Fourth Aerospace Computer Security Application Conference*, Orlando, Florida, December 1988.

[LUNT88] Lunt, T.F. "Automated Audit Trail Analysis and Intrusion Detection: A Survey," *Proceeding of the 11th National Computer Security Conference*, Baltimore, Maryland, October 1988.

[MORG88] Morgenstern, M. "Controlling Logical Inference in Multilevel Database Systems," *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, California, April 1988.

[ORAC89] *Oracle RDBMS Database Administrator's Guide*, Version 6.0, April 1989.

[REED79] Reed, D.P., and R.K. Kanaodia, "Synchronization with Eventcounts and Sequencers," *Communication of the ACM*, 22(2), February 1979.

[ROUG87] Rougeau, P.A., and E.D. Sturms, "The Sybase Secure Dataserver: A Solution to the Multilevel Secure DBMS Problem," *Proceeding of the 10th National Computer Security Conference*, Baltimore, Maryland, September 1987.

[SCHA79] Schaefer, M. "Covert Channel Vulnerabilities in Backend Database Management Architectures," *Air Force Summer Study on Computer Security*, F. Stark Draper Laboratories, Cambridge, Mass., July 1979

[SEB88] Sebring, M.M., E.W. Shellhouse and M.E. Hanna, "Expert Systems in Intrusion Detection: A Case Study," *Proceeding of the 11th National Computer Security Conference*, Baltimore, Maryland, October 1988.

[SMIT88] Smith, G.W. "Inference and Aggregation Security Attach Analysis," Technical Paper, George Manson University, September 1988.

[TCSEC85] *DoD Trusted Computer Systems Evaluation Criteria*, DOD 5200.28-STD, National Computer Security Center, Ft. Meade, MD, December 1985.

[TDI88] *Draft Trusted DBMS Interpretation of the DoD Trusted Computer System Evaluation Criteria*, DoD5200.28-STD, National Computer Security Center, Ft. Meade, MD, November 1988.

[TDI89] *Draft Trusted DBMS Interpretation of Trusted Computer System Evaluation Criteria*, NCSC-TG-021, National Computer Security Center, Ft. Meade, MD, October 1989.

[WHIT74] White, J.C.C. and S.B. Lipner, *Private Communication*, MITRE Corporation, Bedford, Mass., 1974.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. The areas of technical competence include communications, command and control, battle management information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic reliability/maintainability and compatibility.